Get Your Growl ON: Making The BIG Cat Really ROAR!

MacTech Magazine
March • 2006

MACTECH®

The Journal of Macintosh Technology

Make your AppleScript Move at *Lightspeed!*

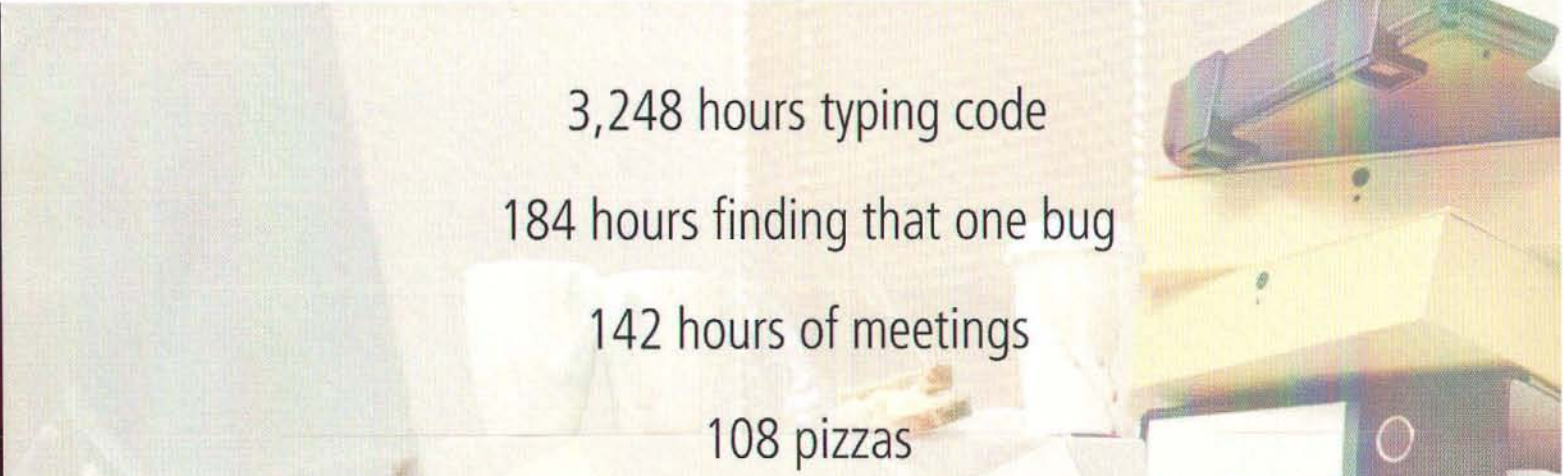
by Ryan Wilcox

Web Surveys: Quick and Easy

Storing and Accessing Data with AppleScript

awk for Data Processing

$8.95 US,
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.



3,248 hours typing code

184 hours finding that one bug

142 hours of meetings

108 pizzas

14 cancelled weekend trips

11 all-nighters

1 call protects it all!

HASP®

SOFTWARE DRM

#1 Choice of Software Developers

The new HASP family of products is the next generation in protection ensuring the highest level of security for your software. It provides an easy set of tools to automatically protect your software and implement new and innovative licensing options.

▸ Strongest anti-piracy solution:

Our powerful, 128-bit AES encryption provides a strong locking mechanism that ensures you get paid for every copy of your software.

ENCRYPTED DATA
PROTECTED SOFTWARE
AES ENCRYPTION ENGINE
DECRYPTED DATA

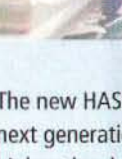
▸ Robust envelope:

Automatically protects Windows executables, DLLs, .Net and Mac applications, securing the weakest link and ensuring your valuable IP remains unreachable. Multiple protection layers provide unequaled security.

▸ Best overall solution as rated by an independent testing lab:

In KeyLabs tests, HASP HL outperformed the competition in security, ease-of-use, flexibility of tools and functionality, and compatibility across platforms and environments. A copy of the full KeyLabs report is available from our Web site at www.aladdin.com/hasp.

Category	Weight	Aladdin HASP HL	Sentinel UltraPro
Security	35%	85.1%	60.9%
Ease-of-use	25%	93.7%	77%
Licensing	20%	75%	75%
Flexibility of tools and functionality	10%	93.3%	71.2%
Compatibility across platforms and environments	10%	93.3%	71%
Overall Evaluation		87.3%	69.8%

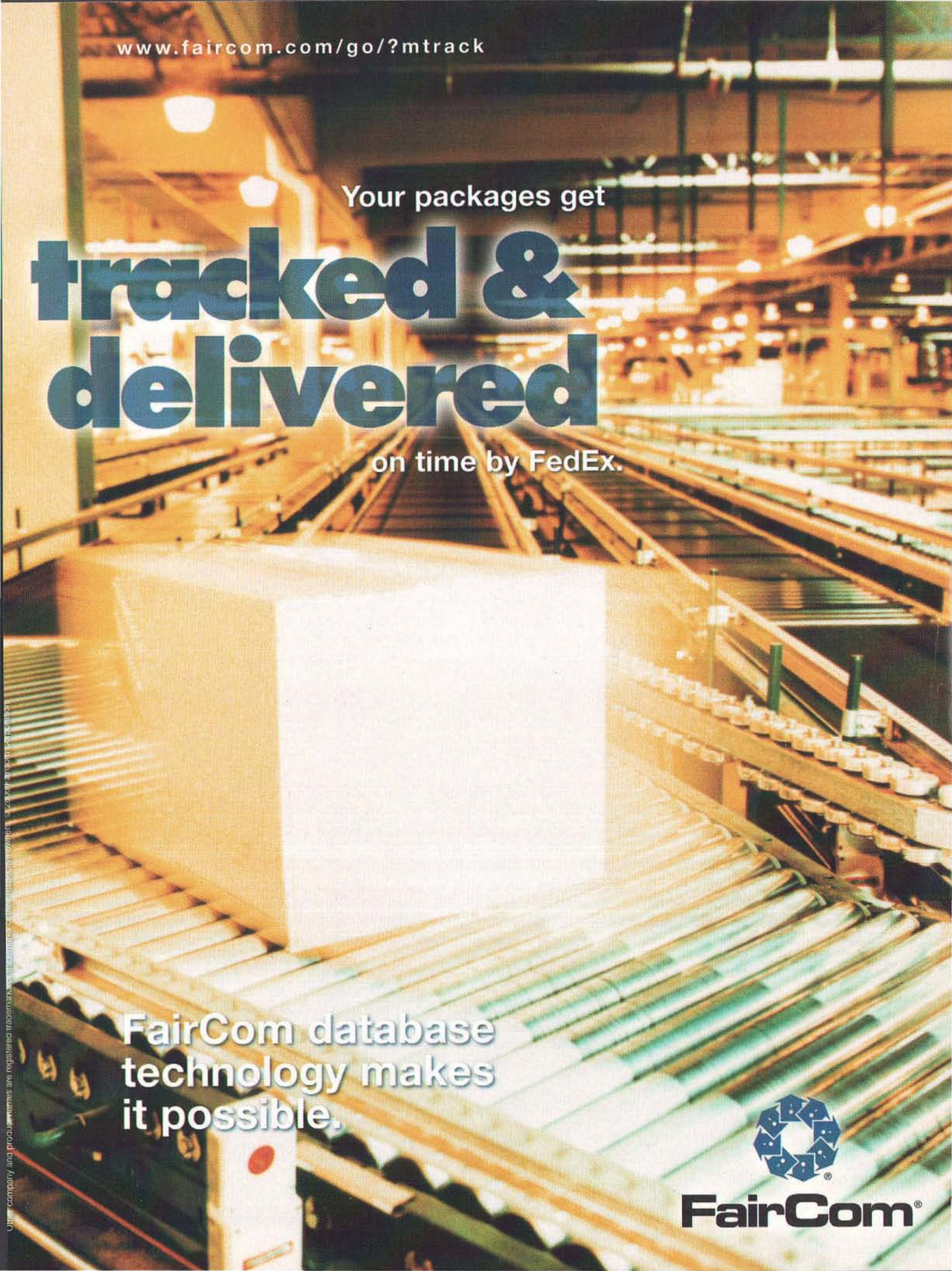


Request a Software Developer Kit today at www.aladdin.com/hasp

Aladdin®

SECURING THE GLOBAL VILLAGE

North America: 1-800-562-2543, 847-818-3800 • UK • Germany • Israel • Benelux • France • Spain • Asia Pacific • Japan


www.faircom.com/go/?mtrack
Your packages get
tracked &
delivered
on time by FedEx.
FairCom database
technology makes
it possible.
FairCom®

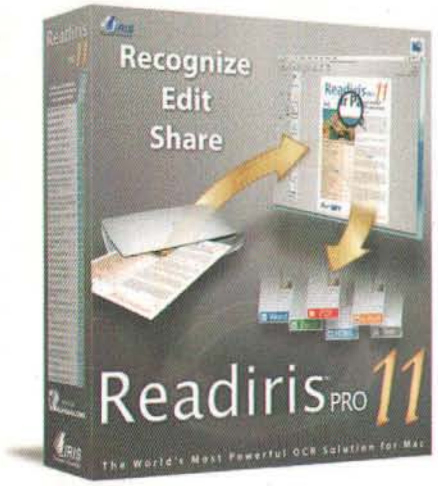
Text Recognition

Readiris™ PRO 11

Mac

The World's Most Powerful OCR Solution for Mac

Readiris Pro 11, the most advanced Optical Character Recognition software for Mac, quickly and easily transforms your paper documents into electronic files you can edit into your favorite application.



Powerful Versatility

Whether you need to recognize a paragraph or a batch of documents, Readiris Pro 11 has all the built-in tools you need. Quickly and accurately convert more document types than ever before.

Create Optimized PDF Archives

Turn any documents into PDF files containing both the image of your original document and the editable text. Reduce your PDF file size and optimize your PDF archiving! The excellent I.R.I.S. recognition quality makes you retrieve the information in a matter of seconds!

Easily Edit and Share

Edit the recognized document in the application you need: in word processors if you want to rework the information, in HTML editors to publish it online, in a new e-mail to immediately share it, etc.

TOP 10 Reasons to Buy

1. **Save Time and Effort with a Fast & Accurate Productivity Tool totally redesigned for the latest Mac OS X versions.**
2. **Create Advanced PDF Files**
3. **New Bar Code Reading capabilities**
4. **Unique Handprinting Recognition engine**
5. **New Configurable Toolbar**
6. **Excellent Format Retention**
7. **New and Easy-to-Use User Interface**
8. **Turn PDF Files into Editable Documents**
9. **Very High Number of Output Formats: RTF, HTML, PDF, TEXT, etc.**
10. **A Multilingual Software**

" Readiris is a smart purchase if you want to create a paperless inventory of your documents. Its accuracy and ease of use make it a terrific buy."

Macdesign magazine (5/5)

MacTech Special

$99 instead of $129, order today at www.irisusa.com/mactech or call 1 800 447 4744

www.irisusa.com
info@irisusa.com

I.R.I.S. *Document to Knowledge™*

J060130-1

T A B L E O F C O N T E N T S

ARTICLES & DEPARTMENTS

Mac In The Shell
awk for Data Processing: The Complementary Pattern Processor to sed.
by Edward Marczak 6

The Source Hound
Get Your Growl ON: Making The BIG Cat Really ROAR!
by Dean Shavit 12

Book Review
Mac OS X Tiger for Unix Geeks
by Mary Norbury-Glaser 24

Random Cocoa: Implementing Five PRNG Algorithms in Cocoa.
by Jose R.C. Cruz 26

AppleScript's Variable Types and You
What you don't know about AppleScript's variable types can hurt you (or slow down your scripts!)
by Ryan Wilcox 42

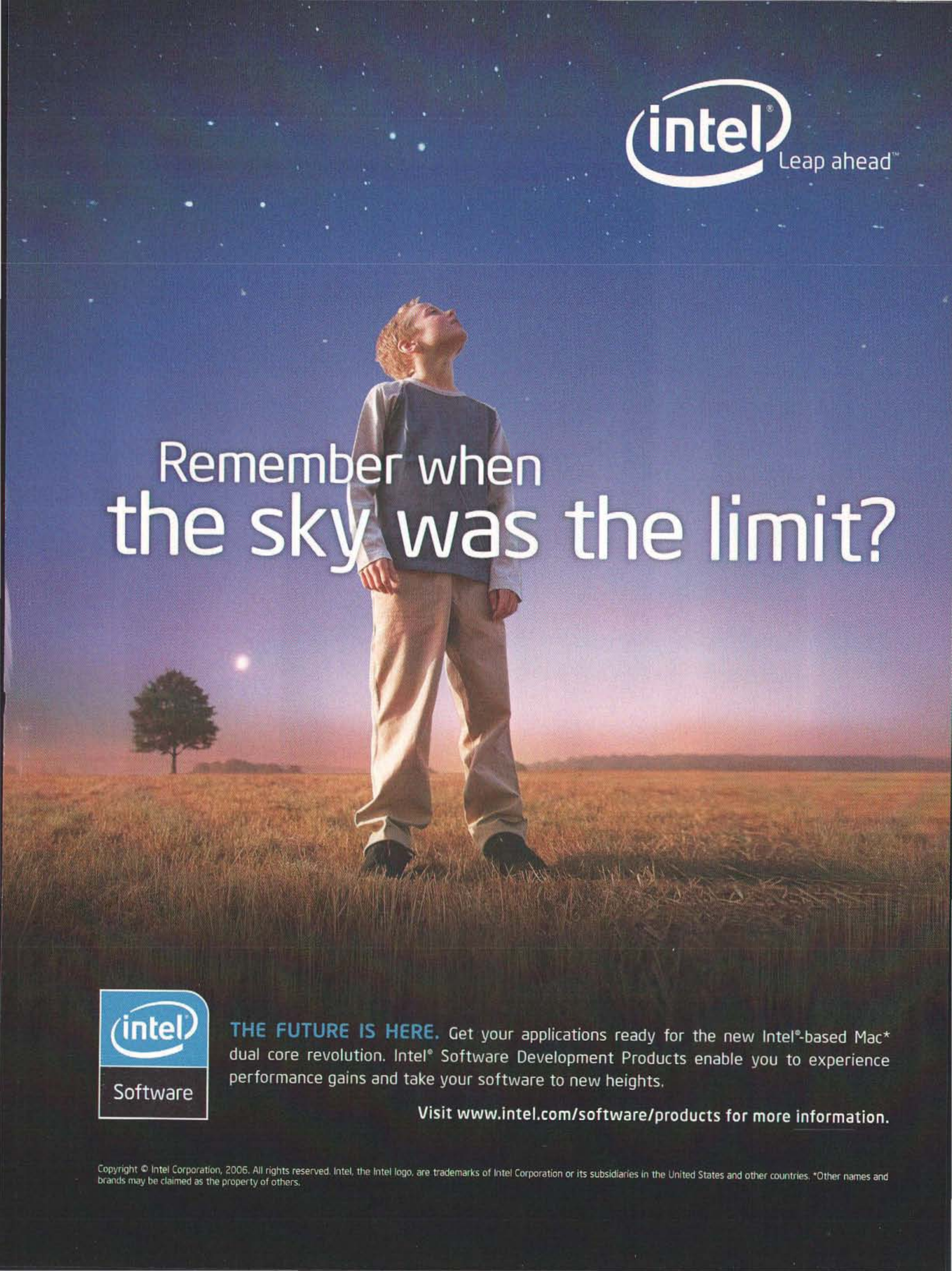
AppleScript Essentials
Storing and Accessing Data with AppleScript
by Ben Waldie 56

Kool Tools
Web Surveys: Quick and Easy 68

Kool Tools
Snapz Pro X 2.0 68

Kool Tools
Klear Screen
The Right Way to Clean Your Screen, iPods, and More 69

In the Trenches
Desktop Systems Engineer and Analyst
by Schoun P. Regan 72



THE FUTURE IS HERE. Get your applications ready for the new Intel®-based Mac* dual core revolution. Intel® Software Development Products enable you to experience performance gains and take your software to new heights.

Visit www.intel.com/software/products for more information.

Communicate With Us

Department E-Mails

Orders, Circulation, & Customer Service
cust_service@mactech.com

Press Releases
press_releases@mactech.com

Ad Sales
adsales@mactech.com

Editorial
editorial@mactech.com
(Authors only, no pr)

Accounting
accounting@mactech.com

Marketing
marketing@mactech.com

General
info@mactech.com

Web Site
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.

MACTECH®
The Journal of Macintosh Technology

A publication of XPLAIN CORPORATION

The Editorial Staff

Publisher & Editor-in-Chief: Neil Ticktin
Associate Publisher & Editor: David Sobsey
Editor-at-Large: Dave Mark
Business Editor: Andrea Sniderman
Production: Dennis Bower
Copy Editor: Marianne Shilpa Jacobie
Staff Writer: Dharmendra Rai

Xplain Corporation Senior Staff

Chief Executive Officer: Neil Ticktin
President: Andrea J. Sniderman
Accounting: Marcie Moriarty
Customer Relations: Susan Pomrantz
Board of Advisors: Steven Geller, Alan Carsrud

Regular Columnists

QuickTime ToolKit: by Tim Monroe
Patch Panel: by John C. Welch
The Source Hound: by Dean Shavit
Reviews/KoolTools: by Michael R. Harvey
AppleScript Essentials: by Ben Waldie
Mac In The Shell: by Ed Marczak

Board of Advisors

Chairman: Dave Mark,
Jordan Dea-Mattson, Steven Geller, Bruce Friedman, and Richard Kimes

Contributing Editors

Michael Brian Bentley, Gordon Garb, Vicki Brown, Chris Kilbourn
Marshall Clow, Rich Morin, Will Porter, Tom Djajadiningrat, Avi Rappoport,
Andrew S. Downs, Cal Simone, Steve Sisak

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

AWK FOR DATA PROCESSING

THE COMPLEMENTARY PATTERN PROCESSOR TO SED.

Sed and awk are typically mentioned in the same sentence. They both have their own strengths and areas where they are most effective. The past few columns have walked though the power of sed, and I hope everyone has put sed into practice. If sed is so great, why do we need awk? sed is a non-interactive editor. It's powerful for unstructured data, and picking out patterns, and making changes in that data. awk excels at pulling, manipulating fields in structured data, and generating output formatted as you specify. You'll encounter both types of data as you work, and now you'll have the best, and most appropriate tools. How can awk help us?

History...Again

When I took one of my very first computer classes, in 7th grade or so, I remember the teacher launching into the history of computing. What?!? History? When are we going to sit down and start typing? Nowadays, I find myself launching into history quite a bit as I write these columns. The benefit is that it frames the present so nicely. A place we couldn't be now without that history. This is a long-winded way of saying I'm going to describe a little bit about the history of awk!

awk appeared in Bell Labs Unix V7 – roughly 1977 - and has been part of the standard distribution since. However, there have been a few revisions and versions of awk. Sometimes, these well-meaning versions have extended awk a little here and there. In 1985, the original authors officially revised the language. I can't possibly cover each and every facet of non-standard awk versions. Since this is MacTech, I'm going to cover Lucent awk, version 20040207, the version distributed with OS X, 10.4. This is the version of awk described in "The AWK Programming Language", 1988, by Al Aho, Peter Weinberger, and Brian Kernighan. (Do we see where the name "awk" comes from now?) Be aware that this version matches the POSIX standard of awk. It does not have every one of the extensions that have shown up over the years.

What is it?

The man page for awk says that it is a "pattern-directed scanning and processing language." The first thing to note is that it is a 'real' programming language, with structure. We've seen flow control and looping in bash and sed before. On a basic level, awk auto-constructs the main loop for you: it loops around each line of input. When it reaches EOF, the loop is broken. Like my Calculus I professor used to drill, "You have to know the rules!" Same goes for any programming language. However, rather than launch into a terse description, let's get right to some examples.

Awk me!

Here's an easy one:

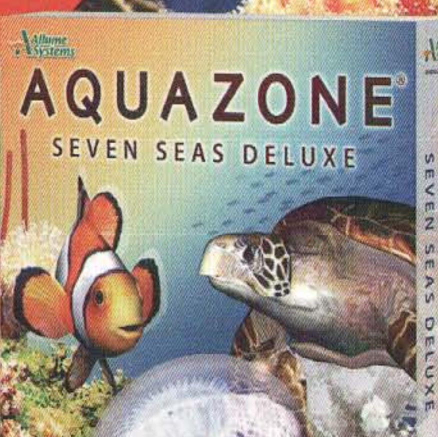
```
$ awk '{print "Got a line"}' some_file.txt
```

This will print "Got a line" for each line in some_file.txt – there's that loop. This script has one action: run a print statement for each line of input. Besides running awk against a file, you can

YOUR ULTIMATE 3D AQUARIUM!

NEW RELEASE!

FORTY BEAUTIFUL FISHES,
TURTLES, JELLIES, & SHARKS!
INTERACTIVE SCREENSAVER
TAP THE GLASS, FEED THE FISH


Allume Systems
AQUAZONE®
SEVEN SEAS DELUXE
AQUAZONE SEVEN SEAS DELUXE
Includes Sharks!
40 fish inside! Have as many onscreen as you like!
Mac

Allume Systems
A Division of Smith Micro Software

A virtual aquarium so vibrant, you'll swear the fish are real! Available from your favorite retailers and catalogs.

Aquazone Seven Seas Deluxe includes 40 fresh and saltwater fish, jellyfish, adult and baby sea turtles, and sharks! Have entire schools of fish on screen swimming through 20 customizable 3D tanks!

also pipe data in. Unlike sed, awk does not print input by default. So, to emulate cat, you could simply:

```
$ awk '{print}' some_file.txt
```

or

```
$ ls -l | awk '{print}'
```

(but using this to emulate cat would be silly). Again, awk really shines when operating on data with a structure. Comma, tab and other delimited formats are ideal – those have obvious structure. However, with enough practice, you'll start to see structure in non-obvious places.

For anyone that really dug into the sed columns, awk's pattern matching will look very familiar:

```
$ ls -l | awk '/pcap/ {print}'
```

We pipe the output of 'ls -l' into awk, where awk will jump into action each time it finds a line with 'pcap' on it. Well, we could have done that with 'ls -l *pcap*', right? Well, yes – but stay with me here. What if we didn't want *all* of the information that comes with 'ls -l'? Or, perhaps, if we wanted to rearrange that info? The output of ls, with the "-l" switch, happens to be very structured. Let's look at a snippet:

```
drwxr-x---   5 marczak  marczak  170 Jan 18 17:07 tmp
-rw-r----   1 marczak  marczak  149 Oct 10 15:17 tw.png
-rw-r----   1 root     marczak 3114 Nov  8 20:00 ts05.pcap
```

awk will refer to each of the columns as ***fields*** – just like a database. The permissions column is field 1, links column is field 2, and so on, up to field 9, in this example, being the file name. If we wanted to rearrange an 'ls' listing, we could use this:

```
$ ls -l | awk '/pcap/ {print $9,$5,$1}'
cramdump.pcap 15151 -rw-r----
dhcp.pcap 16422 -rw-r----
skypecatch.pcap 43421 -rw-r----
ssldump.pcap 26070 -rw-r----
testdump.pcap 12716 -rw-r----
tsnow.pcap 391174 -rw-r----
```

This example combines pattern-matching and the field operator. Again, the output of ls is piped to awk, which only acts when the input line matches "pcap". However, we decide to selectively output only the ninth, fifth and first fields.

Further into the Warren

With sed, we saw that it was good practice to create your script in a separate file – especially if it was a particularly complex script. awk can do the same using the '-f' switch. More conventionally, you may find long awk scripts written like a shell script, utilizing the 'she-bang' notation - #!/usr/bin/awk. Just remember to mark the script executable if you do this.

Another important practice, as pointed out with sed, is to ***comment your script***! With awk, it turns out to be even more important, as you should document the expected input format along with code comments. Any routine that relies on structured data is fragile. When the data isn't perfect, it shatters into a million pieces. So, if you're processing a tab-delimited file, you might start your script with these comments:

```
# thinner.awk
# Remove un-needed data before injecting into mailing
database
# Input: tab delimited file with layout:
# first_name, last_name, phone_num, shoe_size, e-mail, e-
mail2, favorite_color
```

This way, when, three years later, the script stops working the way you'd expect, you can compare the input file against what you need.

awk has some built-in variables that help you move data around. You've seen the field operator - $ - which, I should note, starts at 1. I mean, the first field is actually numbered "1". What happened to programmers counting from zero? The field $0 refers to the entire line of input. A useful built-in that goes along with the field operators is NF.

NF references the number of fields on the current line. A side-effect is that NF will always refer to the last field (or, 'column'). We could rewrite the file listing example above like this:

```
ls -l | awk '/pcap/ {print $NF,$5,$1}'
```

Another important built-in is FS – field separator. Let's look at a very practical OS X use for awk – but we'll need to combine a few concepts to get there. By default, FS is set to a space character. As lines come into awk for processing, it splits up fields by string. Unfortunately, this means that a record reading "Name: Catherine O'Hara" is three fields, not two (of course, it's even worse for "James T. Kirk"). You can leave FS alone, making awk split based on a space character. You can also set FS to be any other single character, such as a comma – obviously useful for a CSV file. Finally, you can use a regexp and match multiple characters as a separator.

In addition to pattern matching to find data to process, awk supports two structures that allow for setup and tear-down (aka pre-processing and post-processing). The BEGIN structure runs before any lines are read in. This is ideal for setting variable states before diving in. The END structure runs after all input is processed, and is naturally useful for summing things up. BEGIN is a perfect place to set FS, although FS can even be changed while the script is running.

So, you're running OS X Server, and want to know who's logged on via AFP. awk to the rescue! Run this:

```
serveradmin command afp:command = getConnectedUsers | awk
'BEGIN {FS = "="} /name/ { print $NF }'
```

The output of serveradmin is fed to awk, which sets FS to the equal sign in a BEGIN structure. This simply splits the line in two, based on the input. Then we go on to look for 'name' records, and print out the last field using NF. Let's say that you just wanted to find out if one particular user is connected. awk will let you test a field for a match with the tilde operator ("~").

So, if we're only interested in finding out if "jane" was connected via afp, we can easily do this:

```
serveradmin command afp:command = getConnectedUsers | awk
'BEGIN {FS = "="} $2 ~ /jane/ { print "Jane is connected!"
}'
```

Of course, you can match any regular expression this way. (didn't I tell you learning regexp would let you rule the universe?) You can invert the tilde match with an exclamation point:

```
awk $2 !~ /barrel/ { print "Not a barrel" }
```

La Règle du Jeu

I mentioned some rules earlier. What are they, and how does that help us? Like sed, awk processes input in a very specific way.

By default, each incoming line is broken into fields, separated by a space. Lines ("records") are separated by a newline. An awk script is a set of pattern matching rules and actions, with the format:

```
pattern {action}
```

Patterns can be one of:

A regular expression
A relational expression
BEGIN
END
A pattern range.

The BEGIN pattern runs its action before the first line of input is read. The END pattern runs its action after the last line of input is read and acted upon.

Some other rules about processing: A missing action defaults to "print". A missing pattern always matches. Program lines are terminated by a semicolon *or* newline. Comments begin with "#" and are not treated as statements. Comments do not need to start at column 1, and will continue until a newline is reached.

If you're thinking, "Hey! awk is pretty powerful ***and*** simple!" you'd be right. Like many Unix utilities it focuses on one thing, and does it really, really well. In some ways, it's only as complex as you make it. Of course, I've only laid out a fraction of awk's abilities. One more before I leave off.

Variables and Equations

Like every programming language, awk supports variables, and operations on those variables. Variables are case sensitive, but do not need to be declared or initialized. Like PHP, this allows variables to be loosely typed, and awk will choose the context automatically. The following examples do what you'd expect:

```
x = 7
y = x+3
a = "Hello, world"
z = $1    # assign the first field to z
print "z = " z
print "a contains " a
print "x = " x
```

Big Security

for the small business

MacWorld **"Best of Show"** Winner, **CRYPTOCard** introduces **5-User Kit for OS X**, the "All-In-One-Box" Authentication Solution that lets your security grow with you.

RYPTO-Server 6.3 is the Mac-centric authentication solution for IT nfrastructures organized around Mac OS X. Once implemented, your lac becomes the centralized authentication and token management ystem for all users, regardless of their computing platform or location.

or Mac Desktops

- Enforced **CRYPTOCard** authentication for desktop access – *Startup, Screen Lock, Sleep*
- Authenticated Fast User Switching
- Internet Connect MSCHAPv2 (Tiger)
- Cisco VPN Client for Mac Integration

Infrastucture Integration

- Open LDAP integration
- Open Directory integration
- Databases (MySQL, Oracle)

Network Integration

- RADIUS-enabled systems – *VPN, Firewall, NAS*
- Web Servers & On-Demand Computing – *Apache, IIS, Citrix*

MACTECH SPECIAL OFFER

5-User Kit *for only* **$249**

* includes 5 tokens, CRYPTO-Server software & 30-day support.

Email **mac@cryptocard.com** and quote **SPMac249-01**.

Promotion can end without notice.

CRYPTOCard Secure Password TECHNOLOGY ®
www.cryptocard.com

Pretty straight-forward. Variables can be used in the pattern portion of a rule. How about a short example?

```
BEGIN { FS=":"; x=0 }
$2 ~ /Miguel/ { x = x  + 1 }
END { print "Miguel appears " x " times in the
data." }
```

This fictitious example adds one to "x" for every time that the second field matches /Miguel/. If I claim that variables don't need to be initialized, why did I in this example? Because the auto-typing can sometimes trip you up. If "x" is ***not*** initialized, and there are ***no*** matches, awk assumes that, due to the context, that "x" is a string. This results in the message, "Miguel appears times in the data." And that's just not very friendly, is it?

In Summary...

Glad I didn't try to rush awk into last month's column. The more that you use both sed and awk, the more you see patterns in data, and tend to go back to these utilities. Despite being created in a time when personal computers (or even larger systems) didn't have their own SQL server running locally, or a powerful spreadsheet program at their disposal, sed and awk still have tremendous usefulness. Next month, I'm going to round out a little more about awk, and tie it into OS X.

Speaking of last month's column, I missed it then, but now realize that it marked "Mac in the Shell's" one-year anniversary! I need to thank David Sobsey, Neil Ticktin, and everyone at the magazine for getting me involved, spurring me along, and keeping me interested. Oh, and Dennis – I loved last month's cover! So, I raise my virtual glass in toast to another great year of MacTech! Cheers!

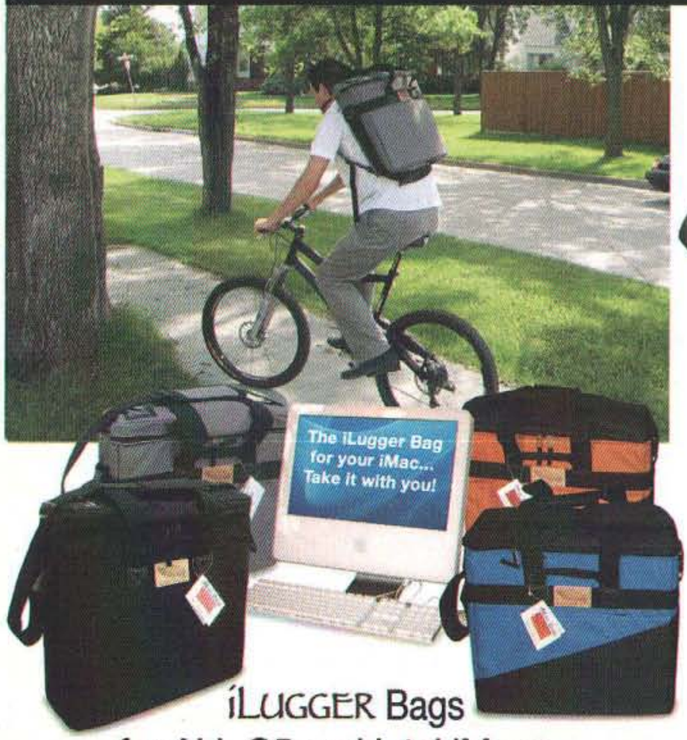
Finally, now that the dust has settled from MacWorld, I do want to say it was a pleasure meeting with many, many MacTech readers! As always, please feel free to comment, suggest and ask questions. See you next month.

MT

About The Author

Ed Marczak owns and operates Radiotope, a technology consulting company. More tech tips at the blog:
http://www.radiotope.com/writing

There are times when you have to take your desktop with you...



Pro Series Bags

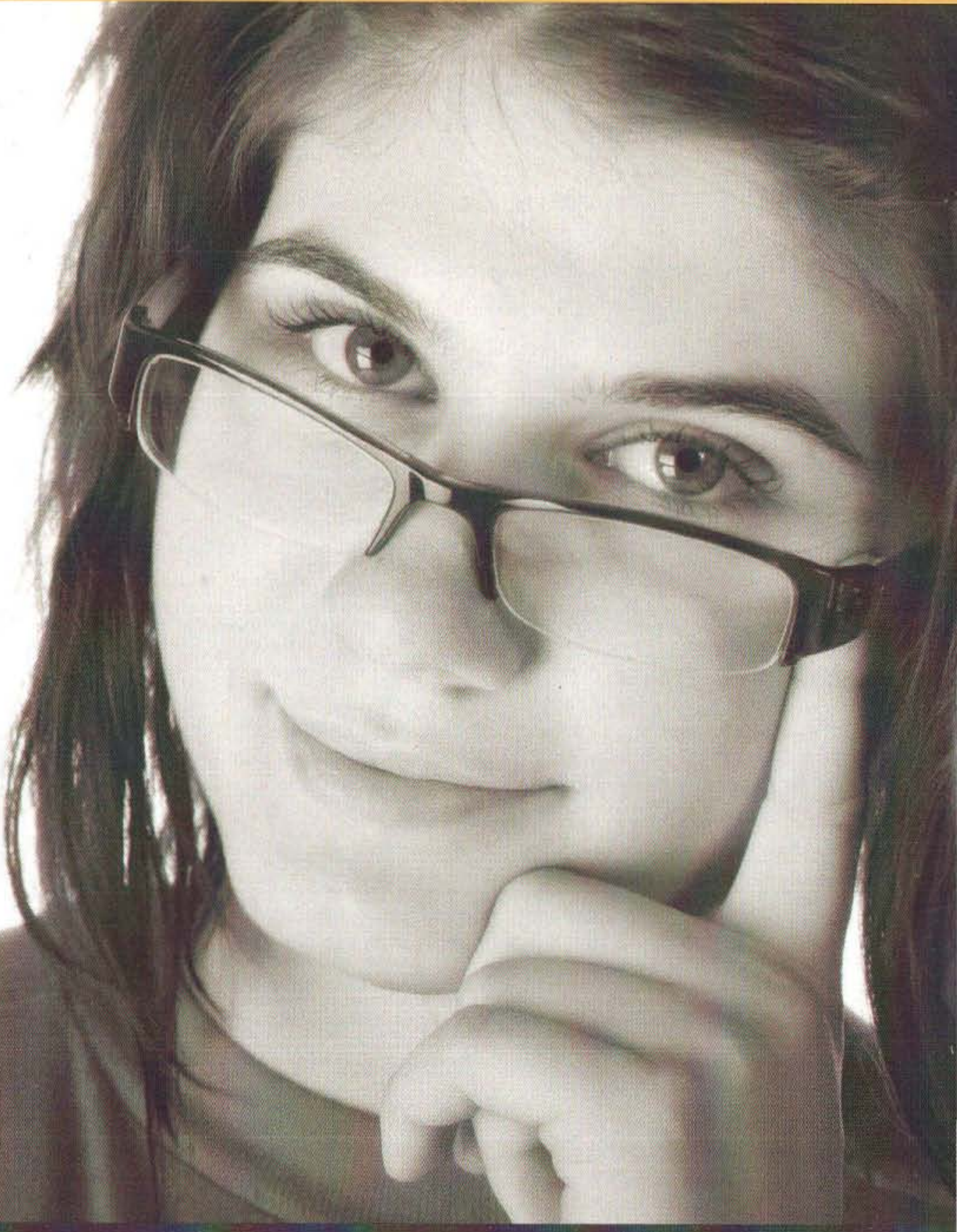
NEW!

iLUGGER Bags
for ALL G5 and Intel iMacs

If you need uncompromising performance ***To Go***
Call us or visit our website

LTA PROJECTS • www.ilugger.com • 877-897-5158 ext 2

'Where were you two hard drives ago?'



Get the fastest software duplication system for Mac OS X working for you.

CopyCatX™ creates lightning fast backups of all your disks – including permissions, icons, placement, and content.

Trying to recover a damaged drive? CopyCatX™ works where other systems fail.

See for yourself.
Visit **www.SubRosaSoft.com** for more information.

SUBROSASOFT

COPYCATX™

SUBROSASOFT.COM

GET YOUR GROWL ON: MAKING THE BIG CAT REALLY ROAR!

Many Mac OS X users tend to think of Open Source Software as "too geeky to be easily used." After four years of life with Mac OS X, many Mac power users and IT pros alike shy away from using the command-line user interface (CLUI) and gravitate toward the GUI (graphical user interface), which is understandable, because mastering the Terminal isn't an absolute necessity to get work done, or even to support others.

Open-Source with Claws

And so the Open Source world has adapted to meet Mac OS X users halfway. We know because Apple keeps reminding us that much of the BSD subsystem of Mac OS X consists of Open Source tools, and we need look no farther than http://www.apple.com/opensource to get Apple's official line:

> Apple believes that using Open Source methodology makes Mac OS X a more robust, secure operating system, as its core components have been subjected to the crucible of peer review for decades. Any problems found with this software can be immediately identified and fixed by Apple and the Open Source community.

So what Apple seems to be telling us is that the "unsexy" parts of Mac OS X, command line tools like "cp" and "cat" and "grep" and "netstat" are the nuts and bolts on which other non-Open Souce goodies of Mac OS X like QuickTime, Spotlight and Exposé depend. In essence, the Open Source parts of Mac OS X, also known as "Darwin" are supposed to be dry and boring, boring and geeky. A quick trip to the source code repository at http://www.opensource.apple.com/darwinsource doesn't do much to alleviate that predilection. Even more high-profile projects like QuickTime Streaming Server and Open Directory don't offer relief in the way of GUI tools, which, for the most part, are ***not*** Open Source, as they are bundled with Mac OS X Server, a commercial product. Even though Open Source super-projects like Fink (http://fink.sourceforge.net) and DarwinPorts (http://darwinports.opendarwin.org) can help us manage the CLUI complexities, and even provide us with comfortable GUIs to do so, it's a pretty safe bet to make the sweeping statement that the vast majority of Apple's involvement in Open Source (I'm not going to count Safari since it's an application, not an OS component) exists within the realm of source code, GCC (the GNU Compiler Collection), and the CLUI. Likewise it is understood that most of the elements making up the "look and feel" of Mac OS X (or the "eye candy" as some like to call it) exist in the realm of Apple's trade secrets and intellectual property.

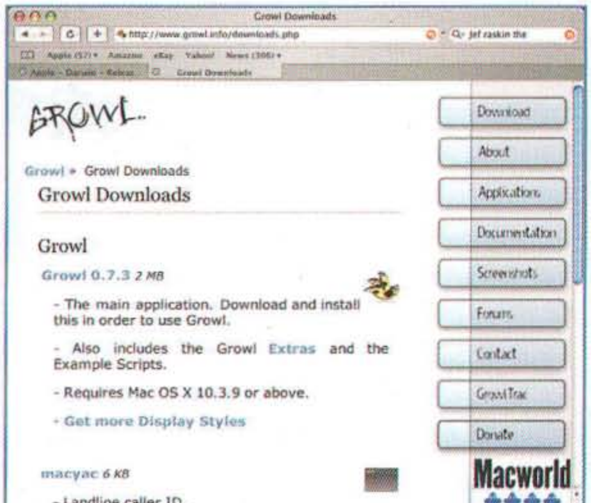
Five years ago, when Mac OS X was in Public Beta, UNIX geeks were busy organizing and finding ways to get Linux and BSD software packages to run on Apple's new fusion of sleek GUI and Open Source. Much of the effort revolved around getting X11, the standard Linux/UNIX windowing system, running on Mac OS X. X11 was about as different from Mac OS X as Microsoft Windows Explorer, with a multitude of themes and desktop managers. Fortunately, Apple released its own distribution of an X11 desktop manager optimized for Mac OS X in 2002, bringing X11 Open Source program much closer to what the typical end-user would consider "usability."

"Make it Mac"

X11, though, with its own keyboard shortcuts that use the control key instead of the standard command key, its menus pinned to the top of each window (like Microsoft Windows) rather than a menu bar at the top of the screen, and its inability to master some basic GUI tricks like exchanging clipboard data with Carbon, Cocoa and Classic applications, hasn't found the following we Open Source advocates had hoped for. The "big kahuna" of Open Source, OpenOffice, ran well under Mac OS X, but its usability (what some would call usefulness) suffered from being an X11 program. In mid-2005, the NeoOffice project (http://www.neooffice.org) finished its amazing NeoOffice/J port of OpenOffice, bringing a polished Open Source alternative to Microsoft Office that runs on Mac OS X without X11. Other Open Source projects, such as Abiword and GIMP can now run in Mac OS X without the benefit of X windows. Whereas such portability seemed like a distant pipe dream five years ago, each Open Source X11 application that can function independently in Mac OS X brings us closer and closer to a roadmap for all of the best Linux and UNIX software to find its way onto the Desktops of end users and power users alike.

Made For Mac

OK, so Open Source is geeky, boring and clunky, though occasionally, with a Herculean effort (the NeoOffice/J project being a prime example), can bear fruit that blossoms into something all Mac OS X users can enjoy. Many projects, like so many of the groupwares (Zimbra collaboration suite being the latest darling) that everyone hopes and wishes might challenge Microsoft Exchange in the Enterprise, may be diamonds in the rough, waiting to be polished a little. But once in a while, seemingly out of the blue, comes an Open Source project so useful and straightforward and clever that it verges on necessity. And when that Open Source project is something ***built for Mac OS X***, it can be a thing of beauty. The Growl project, which lives at http://growl.info is, in this writer's opinion, the perfect example of such a project. Growl simply roars: ***install me, I'm useful***.



Growl » Growl Downloads

Growl Downloads

Growl

Growl 0.7.3 *2 MB*

- The main application. Download and install this in order to use Growl.

- Also includes the Growl Extras and the Example Scripts.

- Requires Mac OS X 10.3.9 or above.

+ Get more Display Styles

macyac *6 KB*

- Landline caller ID.

Figure 1. Growl Download Page.

It's difficult to classify Growl, which is often a clue that an Open Source project is either a completely new way of looking at human interaction with computers or is something that defies categories because of its genius. An example of the former is, The Humane Interface (http://rchi.raskincenter.org), the brainchild of the recently deceased and legendary Jef Raskin, founder of the original Macintosh project at Apple, which may cross into the latter category in time. While the Growl project is an example of something born squarely into the latter: something that defies classification and is simultaneously useful to many people. Before delving into an Open Source project, it's always a good practice to see how the developers see themselves, and the main points of their self-description are positively short and pithy:

First, a bit of marketing:

"Useful notifications that you control"

Then, the crux:

"Growl is a notification system for Mac OS X: it allows applications that support Growl to send you notifications."

Then, the window dressing:

"What are notifications?

Growl includes several display types for notifications.

Notifications are a way for your applications to provide you with new information, without you having to switch from the application you're already in."

Doesn't seem like a whole heck of a lot on first read. But the key really is in the last bit: ". . .without you having to switch from the application you're already in."

Nonplussed

Nonplussed is one of my new favorite words. I think it's a more elegant way of saying something is so surprisingly weak that I sit there with a look of confusion and surprise on my face when beholding the phenomenon for the first time. Such are my feelings regarding the wimpy way the notification system in Mac OS X tries to get someone's attention: it tells the application to ***jump up and down***. When brought to the front, the application displays the pending notification. That's not a terrible idea, but it can be easy to miss, and gives the user absolutely no idea whether the notification is urgent or simply something to dismiss and keep working. Either way, it results in an interruption and requires interaction on the part of the user.

In a similar vein, there's no easy way to have notifications traverse the CLUI barrier and appear in the Finder, though it's not so difficult with a trick or two. Yet the Finder's icon jumping up and down in the dock isn't a really good way of, let's say, reporting that a scripted backup has completed or a `cron` job that checks the status of a RAID mirror has found a problem. Things that are running as system processes or command-line tasks, need to report what they do as well. Since they aren't part of the GUI world, they are left without the type of notifications they deserve, those that would report information in a non-interactive way without having to switch between applications.

Howl for Growl

That's precisely the void Growl aims to fill in Mac OS X. Growl delivers non-invasive, semi-transparent notifications that overlay the Desktop regardless of which application is in the foreground or background, along with a notification title, application icon, and pithy message. Years ago, when I was a kid at summer camp, we used to do this silly thing, pounding our fists on the table, chanting "I scream, you scream, we all scream for ice cream." Now, it's time for all Mac users to rise up, pound their fists on their desks and demand from their IT staff: "I howl, you howl, we all howl to get Growl."

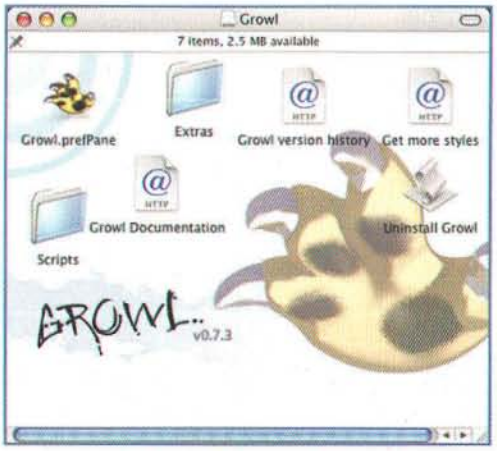
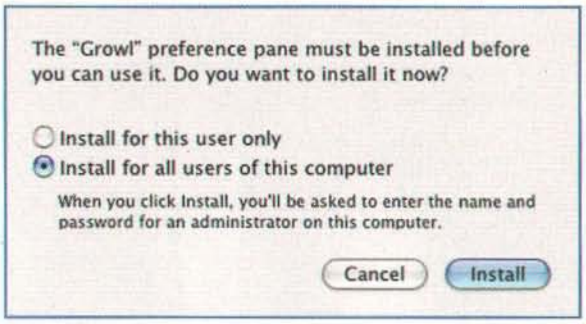


Figure 2. Mounted Growl Disk Image.

Getting Growl is pretty easy. Just download the installer from http://growl.info and double-click on the Growl.prefpane. When asked, decide whether you'd like to install Growl for yourself or for all users of the computer. For all users, you'll need an admin username and password. Even though it's not necessary to be an admin user to install Growl, admin rights are necessary in order to use it as a scripting enhancement, if it's necessary to run script with root privileges, so I recommend installing it for all users of the computer.



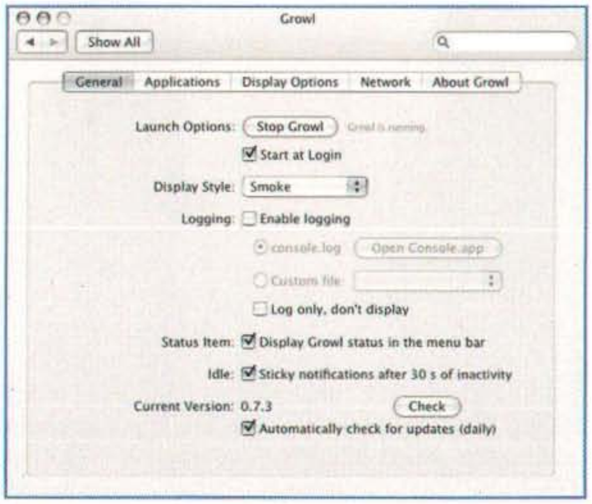
The "Growl" preference pane must be installed before you can use it. Do you want to install it now?

○ Install for this user only
● Install for all users of this computer

When you click Install, you'll be asked to enter the name and password for an administrator on this computer.

Figure 3. Install for One or All Users

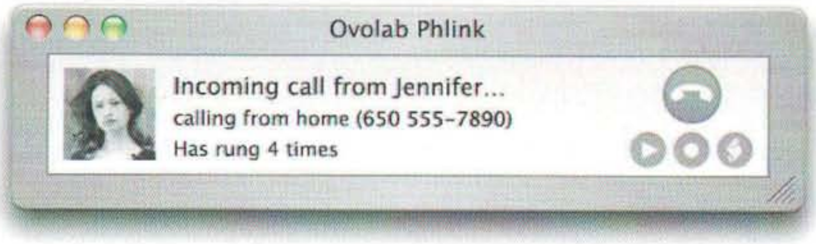
Once installed, the Growl preference pane reveals a number of options, such as the notification style, the applications registered to use Growl, and which notifications they're set to use, as well as a "Stop/Start" Growl item, that either launches or kills the GrowlHelperApp background process. The styles range from the default "Bubbles" to the more serious-looking "Bezel" and the rather in-your-face "Music Video" which places a long black bar across the bottom of the display right where the Dock usually sits. General Growl preferences govern the Growl background process GrowlHelperApp, whether logging is enabled or not, idle and menu bar status icon settings, and whether Growl should automatically check for updates.



Launch Options: Growl is running
☑ Start at Login
Display Style: Smoke
Logging: ☐ Enable logging
● console.log
○ Custom file:
☐ Log only, don't display
Status Item: ☑ Display Growl status in the menu bar
Idle: ☑ Sticky notifications after 30 s of inactivity
Current Version: 0.7.3
☑ Automatically check for updates (daily)

Figure 4. Growl General Settings. Figure 5. Growl Application and Notification Settings.

Look who's talking.


Ovolab Phlink

Incoming call from Jennifer...
calling from home (650 555-7890)
Has rung 4 times

And record all your conversations. And play them back. And a whole lot more.

Ovolab Phlink is the ultimate message center for your Mac. It **answers** phone calls and identifies callers using Caller ID and Apple's Address Book. It greets your friends with personalized messages. It **records** and stores messages on your computer - and even forwards voicemail to **email** as AAC audio attachments. Featuring multiple **voice mailboxes**, high-quality audio, Spotlight searching and fax capabilities, Ovolab Phlink makes your telephone part of the digital hub! And you can fully customize Ovolab Phlink to do exactly what you need, using AppleScript: even set it up to call you back on your cell phone when important clients leave a message. Check it out now at **www.ovolab.com**.

OVO LAB
software for the creative mind

Mac Universal

Macworld MacMinute TidBITS Macworld Expo Superlative macHOME versiontracker

The next tab in the Growl preference pane governs the Growl-savvy applications registered with the Growl notification system and their notifications, which can be toggled between a state of on and off, and whether they are "sticky" meaning that they will stay on screen until receiving a mouse click. Individual display styles are available for each notification, as well as a priority, should there be several notifications queued. One of my absolute favorite Open-Source programs for Mac OS X, is Cyberduck (http://cyberduck.ch), a wonderful FTP/STFP client, made "just for Mac OS X." For long downloads and uploads, it's very useful to have some notifications more informative than a "jumping" duck, and with the "sticky" option, the notice remains on the screen until receiving a mouse click.

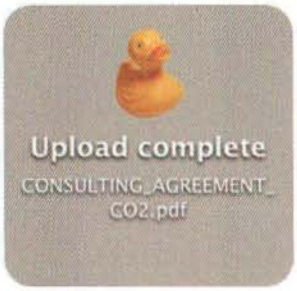
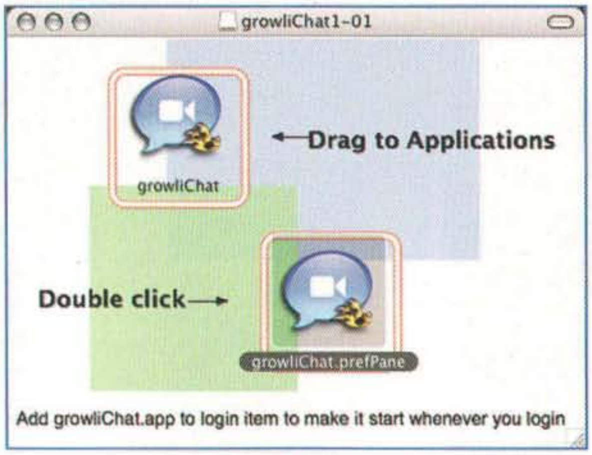


Figure 6. Upload Task Completion Notification.

It's not hard to imagine that useful notifications such as those from Cyberduck would find their way into so many developers' applications. Growl even has recently-added network support, which allows for the relaying of alerts to client machines across a network. It's even possible to set up multiple relays to propagate notifications over a wide area network, though network implementations of Growl are largely unheard of at this point in time. For System Administrators who'd like to send Growl notifications over a network, there's the Growl Perl Module in CPAN (Comprehensive Perl Archive Network) http://search.cpan.org/ ~nmcfarl/Net-Growl-0.99/lib/Net/Growl.pm that can send out Growl notifications ***without*** Growl needing to be installed on the originating host, opening up the door to notifications that might come from a Linux or Windows box as well. As a matter of fact, with some cooperation, it's not a stretch to imagine Growl and a network monitor like Nagios (http://www.nagios.org) complementing each other to form a comprehensive solution for local and remote alerts via web server, pager, email and the Desktop.

Growl, Who's There?

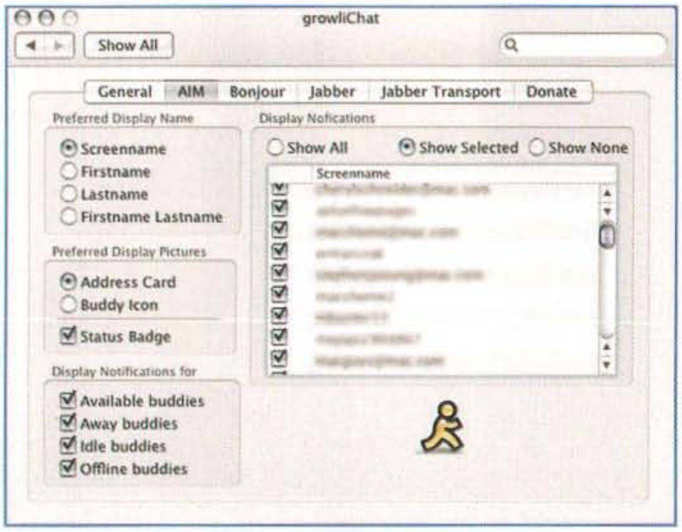
Perhaps the most common use for Growl is an application which many Mac users depend on for daily interaction and communication, yet suffer from the "jumping" icon syndrome: iChat. As people on an iChat buddy list come online or go offline, the user gets background sound effects. If a chat is initiated, and iChat is in the background, iChat ***jumps***, it doesn't say who is inviting you to chat nor will it tell you who is available, or who would like your attention. Wouldn't it be nice if it did? Wouldn't it be nice to see the incoming status messages from a backgrounded chat session without brining iChat to the front? Wouldn't it be nice to know exactly who is available or not available with a translucent status message rather than a ***whoosh*** sound effect, that says "someone is either coming or going, I don't know who. . ." Well, that's precisely what the free enhancement growliChat (http://www.growliChat .com) brings to Mac OS X. Installing growliChat is a piece of cake, just download it and double click on the disk image (.dmg), then double-click the prefPane to install it and move the application to /Applications.



Add growliChat.app to login item to make it start whenever you login

Figure 7. Installing GrowliChat .

Once installed, and with growliChat running, it's necessary to specify the desired notification behavior. The default is usually good enough, though it's possible to turn notifications on or off for specific buddies. Like Growl, growliChat is configured via a preference pane, with a few tabs of options for each major form of iChat trasport: AIM, Bonjour, and Jabber, making it a suitable enhancement for the new iChat service bundled with Mac OS X Tiger Server.



Preferred Display Name: [✓ Screenname] Firstname Lastname Firstname Lastname

Preferred Display Pictures: [✓ Address Card] Buddy Icon

[✓ Status Badge]

Display Nofications: Show All [✓ Show Selected] Show None

Display Notifications for: [✓ Available buddies] [✓ Away buddies] [✓ Idle buddies] [✓ Offline buddies]

Figure 8. GrowliChat AIM Notification Settings.

Once everything's configured accordingly, and growliChat is running and registered with Growl, the fun

Innovations by InterSystems

Messaging, plus...

Composite Application Development, plus...

Business Process Orchestration, plus...

Business Activity Monitoring

Integrate Your Enterprise. Ensemble Enables EAI, BAM, BPM, And All That Jazz.

Ensemble is the first product that enables fast integration and rapid development. This unique platform combines the capabilities of an integration server, data server, application server, and portal development software.

What makes this possible is an innovation by InterSystems: We developed Ensemble as a *single* technology stack, not a stitched-together suite of separate components. With a unified environment for integration, development, and management, Ensemble dramatically reduces time-to-solution.

We back these claims with this money-back guarantee: *For up to one year after you purchase Ensemble, if you are unhappy for any reason, we'll refund 100% of your license fee.** We are InterSystems, a global software company with a track record of innovation for more than 25 years.

InterSystems
ENSEMBLE™

Request a FREE proof-of-concept project at **www.InterSystems.com/Ensemble12HH**

* Read about our money-back guarantee at the web page shown above.

begins! Now, instead of the *whoosh* sound when a buddy comes online, and having to bring iChat to the foreground to see who may have become available or away, the user's greeted with the following notification:

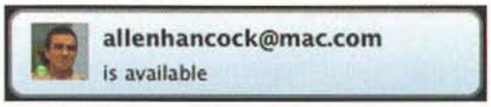

allenhancock@mac.com
is available

Figure 9. GrowliChat Buddy Available Notification

I can't begin to gush over how useful this is compared to a *whoosh* sound. The notification presents the screenname, first name, or full name of the buddy changing status, the status (available) and even a picture (or icon) of the buddy as if the name weren't enough! Since I installed growliChat, I no longer find myself bringing my buddy list window to the front to check out who's left or arrived. So powerful and yet so simple. Kind of like the spirit of the Macintosh itself.

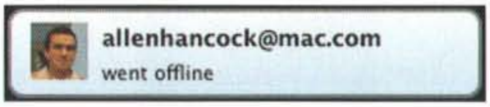

allenhancock@mac.com
went offline

Figure 10. GrowliChat Buddy Offline Notification.

If You Build It, They Will Come

One of the classic mythical ways to make a fortune is to "build a better mousetrap." That whole notion is predicated on the fact, that, everyone has a problem with mice, which Apple has now seemed to address with the addition of the Mighty Mouse to its product lineup. But in the Mac OS X world, people have an issue with application notifications, even if they don't realize it. Others have called Growl an enabler with "multiplier" capabilities that could possibly enrich the entire Mac OS X Software landscape. Today, about 150 applications in 13 categories sport Growl support, from Powermail (http://ctmdev.com/) to the Shiira Web Browser (http://hmdt-web.net/shiira), that shares the Webkit engine with Apple's own Safari to the FTP Clients Cyberduck and Transmit (http://panic.com/transmit). Personally, I can't imagine any developer working on an application that benefited from notifications ***not*** considering using Growl support. It just doesn't make sense to roll your own. Hear that, Roxio?

When faced with the task of notifying users of my own application, Mac HelpMate, (http://www.macworkshops.com/machelpmate), that a scheduled maintenance task had completed, I wasn't very thrilled with having a dialog pop up in the Finder, to trigger a ***leaping*** Finder icon in the dock, forcing the user to bring the Finder to the front to get the message. Does that sound like a repetitive and familiar complaint? Sure! So, I decided that, if a Mac HelpMate user wanted to install Growl, then I'd support Growl for the notifications instead.

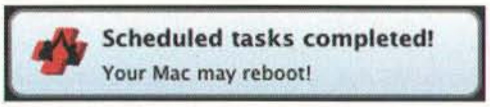

Scheduled tasks completed!
Your Mac may reboot!

Figure 11. Mac HelpMate Scheduled Task Completion.

Installing Growl also installs the Growl.Framework for Cocoa Developers in /Library/Frameworks. However, since I'm not a Cocoa developer (Although I aspire to be at some point in the future, for now Mac HelpMate is an AppleScript Studio effort), I needed a way to hook up with Growl notifications rather than by using Objective C. Fortunately for me, Growl has some rather easily accessed support for AppleScript. All it takes is a little imagination to add fancy notifications to even the simplest of applications.

Hear that, Apple? There's even a certain crowd of independent developers and their associates who would advocate Apple adopting Growl or choosing to bundle it with Mac OS X. Although Apple has its own translucent bezel notifications, the most conspicuous of which, are the translucent "eject" icon or "volume" icons, that appear over the Desktop, there's not a readily available Framework available for Developers to use. Even though Apple has a track record of imitating popular eye-catching technologies such as Watson (for Sherlock) then Konfabulator for Dashboard, Growl has already gained so much momentum and is complex enough (network support is a perfect example) that Apple would be hard pressed to clone, bundle or support Growl for the typical Mac OS X user. For that reason, my money is on Growl remaining a growlingly popular application most Mac users will never know about (sniffle), unless we, their System Administrators, Support Pros and Consultants, give it to them.

If You Build It, You Can Growl

Those of you who read my column certainly are familiar with the bio and blurb I use to close out each one, but this time, I want to do it a different way. This time, I'm going to sign off using a Growl notification. Let's start with a simple AppleScript application designed to do two things: first, register itself as a Growl application, second, actually send a notification from the application to the Desktop via the GrowlHelpApp process. I'm going to use the sample ApplesScript code from the Growl site available at http://growl.info/documentation/applescript-support.php The simple AppleScript application is going to be called "Authorbio."

Step 1: open up the "Script Editor" application in /Applications/AppleScript and copy and paste the sample AppleScript code into the new script window

Premium

Small Business Management & Accounting

Software

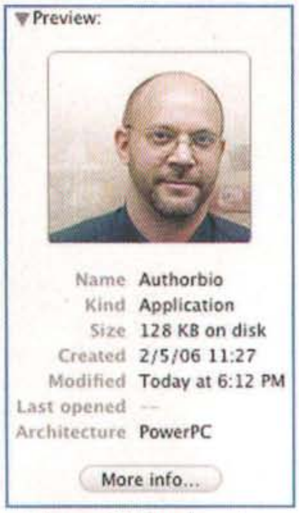
Mind Your Own Business. Smarter.

800.322.MYOB (6962)

www.myob-us.com

Step 2: modify the AppleScript with your desired Application name, and your desired notifications

Step 3: use the very cool Open Source utility img2icns from www.rknet.it/program/img2icns to create a folder from a picture of yourself and then cut and past the icon on to your Application when your finished, like so:



▼Preview:

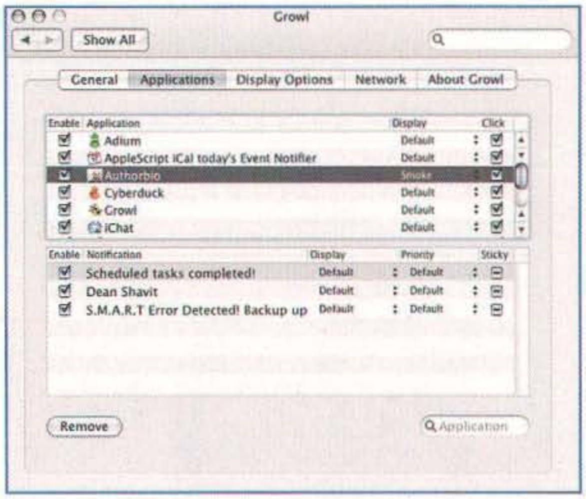
Name	Authorbio
Kind	Application
Size	128 KB on disk
Created	2/5/06 11:27
Modified	Today at 6:12 PM
Last opened	--
Architecture	PowerPC

More info...

Figure 12. Custom Icon from Photograph.

Step 4. Save the AppleScript as an Application, and cut and paste the icon of yourself on the Authorbio Application so that when you "sign out" using the Growl notification of your choice, your face is showing next to the alert—cool!

Step 5: After initially registering the Application in the Growl preference pane in System Preferences, make sure you've selected an alert style capable of displaying a lot of text. For my own purposes, I prefer the "smoke" style.



Growl

Show All | General | Applications | Display Options | Network | About Growl

Enable	Application	Display	Click
✓	Adium	Default	✓
✓	AppleScript iCal today's Event Notifier	Default	✓
✓	Authorbio	Smoke	✓
✓	Cyberduck	Default	✓
✓	Growl	Default	✓
✓	iChat	Default	✓

Enable	Notification	Display	Priority	Sticky
✓	Scheduled tasks completed!	Default	Default	⊟
✓	Dean Shavit	Default	Default	⊟
✓	S.M.A.R.T Error Detected! Backup up	Default	Default	⊟

Remove | Application

Figure 13. Select Appropriate Alert Style.

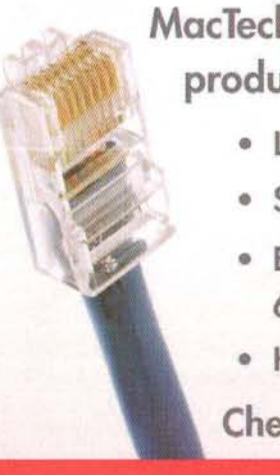
Here's the AppleScript code of my Authorbio.app Application. You can download the entire project from my site, http://www.themacheldesk.com and customize it for your own use. There's even an example on how to use osascript to run the notification from the Terminal, so you can use it to notify you when cron jobs or launchd jobs complete!

–present the choice in a dialog:

MACTECH® Cable Supplies

Looking for a great selection of cables?

MacTech has selected a quality vendor, with quality products for your cable needs. Benefits include:



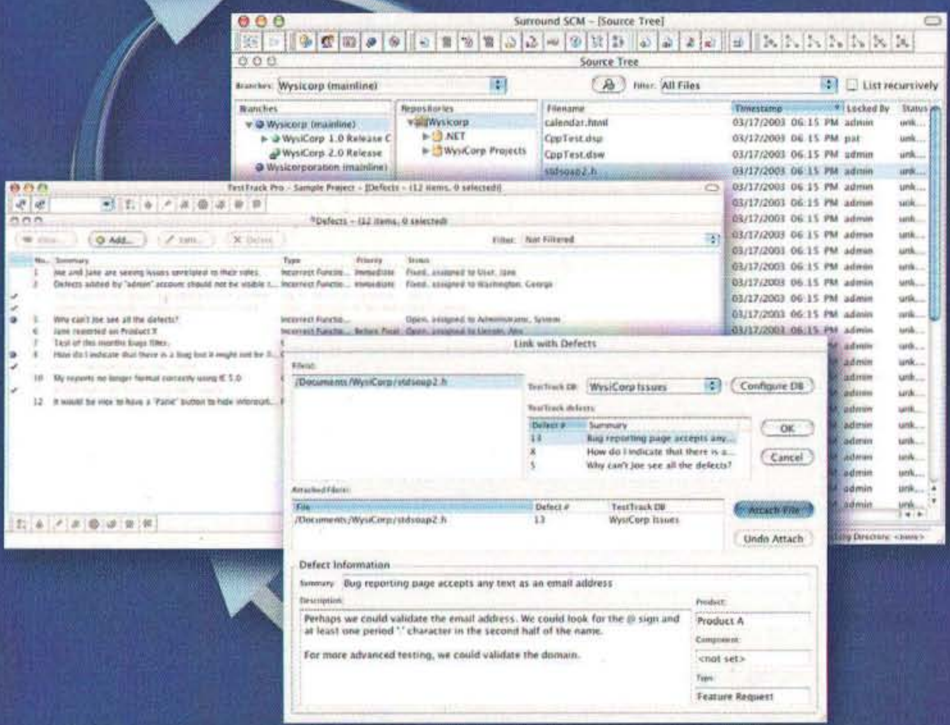
- Lifetime warranty on cables
- Same day shipping if you order by 4 PM ET
- Express shipping orders can be ordered up to 6:30 PM ET for same day shipping
- Huge selection - 3500+ items and growing

Check out the full selection at the MacTech Cable Store

http://www.mactech.com/cables

Complete Source Control and Defect Management for Mac OS X

Seapine Software™
changing the world of software development



Surround SCM – [Source Tree]

Source Tree

Branches: Wysicorp (mainline) — Filter: All Files — List recursively

Branches: Wysicorp (mainline); WysiCorp 1.0 Release C; WysiCorp 2.0 Release; Wysicorporation (mainline)

Repositories: Wysicorp; .NET; WysiCorp Projects

Filename	Timestamp	Locked By	Status
calendar.html	03/17/2003 06:15 PM	admin	unk...
CppTest.dsp	03/17/2003 06:15 PM	pat	unk...
CppTest.dsw	03/17/2003 06:15 PM	admin	unk...
stdsoap2.h	03/17/2003 06:15 PM	admin	unk...

TestTrack Pro - Sample Project - [Defects - (12 items, 0 selected)]

*Defects - (12 items, 0 selected)

Filter: Not Filtered

No.	Summary	Type	Priority	Status
1	Joe and Jane are seeing issues unrelated to their roles.	Incorrect Functio...	Immediate	Fixed, assigned to User, Jane
2	Defects added by "admin" account should not be visible t...	Incorrect Functio...	Immediate	Fixed, assigned to Washington, George
5	Why can't Joe see all the defects?	Incorrect Functio...		Open, assigned to Administrator, System
6	Jane reported on Product X	Incorrect Functio...	Before Final	Open, assigned to Lincoln, Abe
7	Test of this months bugs filter.			
8	How do I indicate that there is a bug but it might not be ...			
10	My reports no longer format correctly using IE 5.0			
12	It would be nice to have a "Panic" button to hide informati...			

Link with Defects

Files: /Documents/WysiCorp/stdsoap2.h

TestTrack DB: WysiCorp Issues — Configure DB

TestTrack defects:

Defect #	Summary
13	Bug reporting page accepts any...
8	How do I indicate that there is a...
5	Why can't Joe see all the defects?

OK — Cancel

Attached Files:

File	Defect #	TestTrack DB
/Documents/WysiCorp/stdsoap2.h	13	WysiCorp Issues

Attach File — Undo Attach

Defect Information

Summary: Bug reporting page accepts any text as an email address

Description: Perhaps we could validate the email address. We could look for the @ sign and at least one period '.' character in the second half of the name.

For more advanced testing, we could validate the domain.

Product: Product A

Component: <not set>

Type: Feature Request

Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- **New!** Full Unicode support maintains international characters across various languages, ensuring data is not lost or misinterpreted
- Advanced branching simplifies managing multiple versions of your products
- Fast and secure remote access to your source files and defects — work from anywhere
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine Surround SCM

Seapine TestTrack PRO

software development 13th annual product excellence award JOLT

Download Surround SCM and TestTrack Pro at www.seapine.com/mac or call 1-888-683-6456

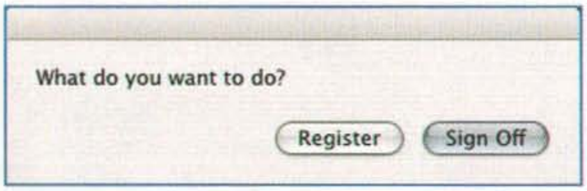
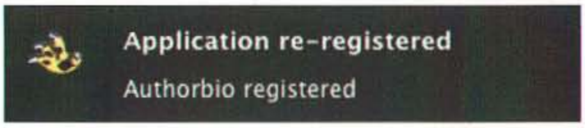

What do you want to do?

Figure 14. Authorbio.app User Interface.

```
set mygrowl to display dialog ¬
  "What do you want to do?" buttons ["Register",
"Sign Off"] ¬
  default button 2
set grrr to button returned of mygrowl

--if "register" set up notitications and register
with Growl

if the grrr is equal to "register" then
  tell application "GrowlHelperApp"

    -- Make a list of all the notification types
    -- that this script will ever send:
    set the allNotificationsList to ¬
      {"Scheduled tasks completed!", "Dean Shavit",
¬
        "S.M.A.R.T Error Detected! Backup up your
data ASAP!"}

    -- Make a list of the notifications
    -- that will be enabled by default.
    -- Those not enabled by default can be enabled
later
    -- in the 'Applications' tab of the growl
prefpane.
    set the enabledNotificationsList to ¬
      {"Scheduled tasks completed!", "Dean Shavit",
¬
        "S.M.A.R.T Error Detected! Backup up your
data ASAP!"}

    -- Register our script with growl.
    -- You can optionally (as here) set a default
icon
    -- for this script's notifications.
    register as application ¬
      "Authorbio" all notifications
allNotificationsList default notifications ¬
      enabledNotificationsList icon of application
"Authorbio"
  end tell

  --if "sign off" then say goodbye to this issue of
MacTech

else if grrr is equal to "Sign Off" then
  tell application "GrowlHelperApp"
    -- Send a Notification...
    notify with name "Dean Shavit" title "Dean
Shavit, a.k.a Sourcehound" description "is an ACSA
(Apple Certified System Administrator) who loves
Open-Source and freeware solutions for Mac OS X.
During the day, he is a partner at MOST Training &
Consulting in Chicago, where he trains system
administrators in Mac OS X and Mac OS X Server,
helping his customers get the best ROI possible
from their computer investment while writing for
his own website, www.theMachelpdesk.com. Recently,
he became the father of an application: the Mac
HelpMate troubleshooting tool, available at
www.Machelpmate.com. If you have questions or
comments you can contact him:
dean@Macworkshops.com" application name "Authorbio"
with sticky
  end tell
end if
```

OK, let's run the Application. First, we'll need to choose to register the application so it shows up as a "known" alerting process in the Growl System Preference. Launch the Authorbio, and click the "register" button:

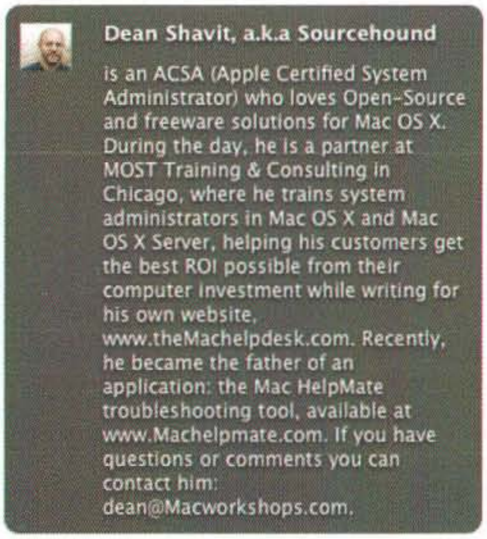

Application re-registered

Authorbio registered

Figure 15. Application Registration

Now, let's try it again, except this time, I'll click the "Sign off Button," and make my grand exit. I'm unsure what I'm going to write about in my next column, but this is such an exciting time to be a Mac user, the world's a veritable oyster. So, my good friends, *ciao* for now. . . ***Signing off***...

MT


Dean Shavit, a.k.a Sourcehound

is an ACSA (Apple Certified System Administrator) who loves Open-Source and freeware solutions for Mac OS X. During the day, he is a partner at MOST Training & Consulting in Chicago, where he trains system administrators in Mac OS X and Mac OS X Server, helping his customers get the best ROI possible from their computer investment while writing for his own website, www.theMachelpdesk.com. Recently, he became the father of an application: the Mac HelpMate troubleshooting tool, available at www.Machelpmate.com. If you have questions or comments you can contact him: dean@Macworkshops.com.

script happens

MacScripter

Your World Scripted!

http://macscripter.net

Intego VirusBarrier X4

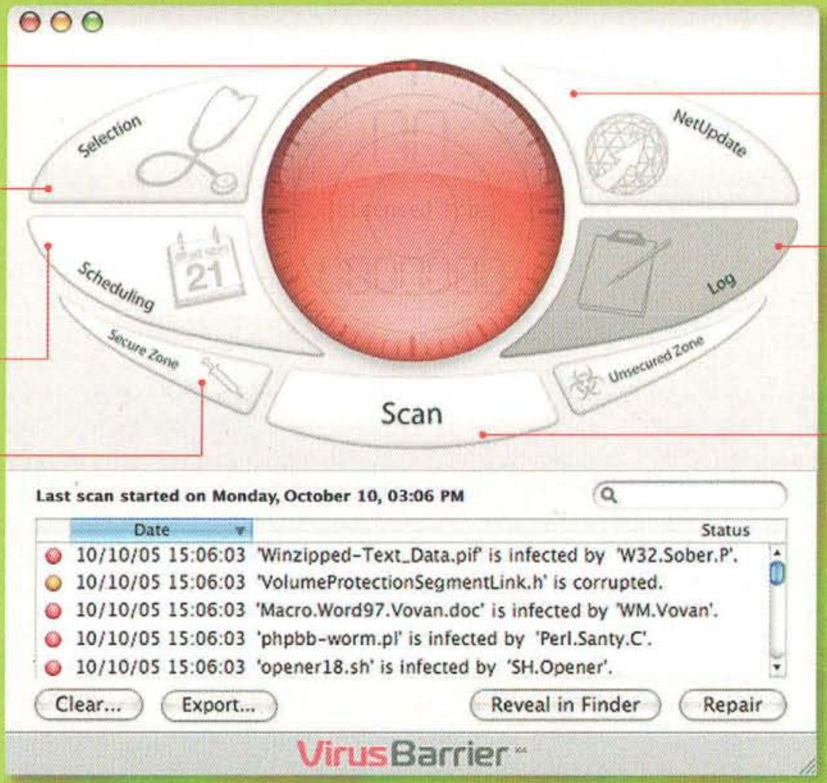
The acclaimed antivirus program for the Mac

Multi-function Orb
The change in color of the Orb indicates the presence of an infected or damaged file.

Selection
Choose the data to be scanned in the dialog box with a simple click.

Scheduling
Schedule regular or specific scans of your volumes and view them in Apple's iCal.

Reinforced security zone
Decide for yourself the best level of security for your files and folders.



Last scan started on Monday, October 10, 03:06 PM

```
Date                                                                      Status
10/10/05 15:06:03 'Winzipped-Text_Data.pif' is infected by 'W32.Sober.P'.
10/10/05 15:06:03 'VolumeProtectionSegmentLink.h' is corrupted.
10/10/05 15:06:03 'Macro.Word97.Vovan.doc' is infected by 'WM.Vovan'.
10/10/05 15:06:03 'phpbb-worm.pl' is infected by 'Perl.Santy.C'.
10/10/05 15:06:03 'opener18.sh' is infected by 'SH.Opener'.
```

NetUpdate X4
Automatically or manually check the availability of updates or virus definitions.

Complete log
Display all analysis logs of your volumes and files with a single click.

Analysis
Turbo Mode technology analyzes your data up to 40 times faster.

Protects against the Oompa-Loompa (OSX/Oomp-A or Leap.A)

For Macs with PowerPC or Intel processors

Intego Widgets
Discover, under Tiger, new Intego Widgets informing you of the status of your protection, the availability of updates, and scheduled events.

Repairs your infected documents
VirusBarrier X4 alerts you to the presence of viruses and repairs damaged or corrupted files.

Your updates with NetUpdate X4
Easily schedule, check and install new virus definitions and updates for all Intego software installed on your Mac.

Macworld ★★★★ Editor's Choice Best antivirus for Mac, March 2005
VirusBarrier X4

Main features of VirusBarrier X4

- **Detects** and **eliminates** all known viruses
- **Repairs** infected files
- **Turbo Mode** technology (up to 40 times faster)
- **Simple, effective** and **non-intrusive**
- New dynamic **resizable interface**
- **Scans** incoming and outgoing **e-mail**
- New **Intego menu**
- **VirusBarrier X4 Widget** and Intego Widget
- **Scheduled** scans
- **Detailed** log
- Reinforced **security zone**
- **Heuristic** and **behavioral** analysis
- **Blocks** virus execution
- New **multi-function Orb**
- New **alert management**
- File **analysis** by the Intego Virus Monitoring Center
- File creation and modification **detection**
- **Temporary** antivirus deactivation
- Easy **integration** with other Intego programs
- Management of **compressed files**
- **Contextual Menu**
- **Updates** via NetUpdate X4

Mac Universal

Apple Store MacMall NAVARRE CORPORATION Fry's J&R MUSIC WORLD www.intego.com

Intego • 500 North Capital of Texas Hway, Suite 8-150 - Austin, TX 78746 • Tel (512) 637-0700 • Fax (512) 637-0701 • sales@intego.com

intego
we **protect** your world

Book Review

by Mary Norbury-Glaser

MAC OS X TIGER FOR UNIX GEEKS

Demystifying the Geekier Side of Mac OS X
Mac OS X Tiger for Unix Geeks
O'REILLY
Brian Jepson & Ernest E. Rothman

Mac OS X Tiger for Unix Geeks, published by O'Reilly, is a book for Unix sysadmins and developers who are interested in discovering Mac OS X's underlying Unix roots. The authors, Brian Jepson and Ernest E. Rothman, assume the reader has a solid familiarity with Unix fundamentals.

The book is divided into four instructional parts (Getting Around, Building Applications, Working With Packages, and Serving and System Management) and a final section devoted to two appendicxes (Mac OS X GUI Primer and Mac OS X's Unix Development Tools).

Part I is an introduction to Unix on Mac OS X in ten chapters and is so extensive that it constitutes half the book. Chapter 1 introduces differences between the Terminal app and xterm and xterm-like applications specific to systems running X Windows, how to customize the terminal, and working with files and directories. Chapter 2 covers Spotlight and searching metadata from the command line using `mdfind`. The next chapter deals with file transfer issues that arise from moving data between different file systems. This is a particularly helpful section on understanding the consequences of transferring large numbers of files between foreign operating systems. The authors also list directories in tabular format: `/`, `/etc`, `/System/Library`, `/Library`, `/var`, and `/dev`.

There are some distinct startup differences between traditional Unix systems and Mac OS X Tiger, detailed in and the boot process is detailed in Chapter 4. In fact, there are changes between Tiger and pre-Tiger systems and this chapter presents an outline of the differences and consequences very nicely. A logical consequence of this discussion is how to add startup items and schedule tasks and the authors illustrate this with a very useful MySQL startup script.

Chapter 5, Directory Services, gives a brief overview of understanding and configuring Open Directory, and Mac OS X's version of Directory Services. There is a very helpful section on working with passwords and the authors demonstrate four NetInfo/Directory Services utilities: `dscl`, `nireport`, `nidump`, and `niload`.

Printing through both the GUI and through the included Unix tools (CUPS and Gimp-Print) is the topic of Chapter 6. This chapter is well stocked with screenshots and is a quick read.

Chapter 7 goes over Apple's X11 implementation of the X Window System: installing, running full-screen or rootless, customizing preferences, connecting to other X Window systems using `ssh` with X11 forwarding, using osx2x to share a keyboard and mouse between systems, and how to use `vnc`.

The topic in of Chapter 8 is multimedia and discusses burning CDs, using the open source MPlayer for audio/video playback, the Gimp for image editing, and Blender for 3D modeling.

In the next chapter, the authors include third-party tools and applications: virtual desktop implementations, `ssh` GUI frontends, and open source typesetting, mathematical and statistical apps (TeX and R), and office suites (OpenOffice and NeoOffice/J). The information on TeX and R is particularly well documented, due no doubt due to Dr. Rothman's scholarly pursuits. Part I ends with a chapter on dual booting various flavors of Linux (Yellow Dog, Gentoo, Ubuntu, etc.) with Mac OS X, using emulators (VirtualPC) as well as running Mac OS X on x386 machines using PearPC.

Part II, Building Applications, is for those developers and Unix/Linux power users who are comfortable with compiling and linking source code. There are plenty of subtle differences between the Unix and Mac OS X compilers, and the authors provide ample instruction on how to work with Apple's version of the GCC (GNU Compiler Collection). The authors provide a complete discussion of compiling source on Mac OS X and include frameworks, architectural issues (AltiVec, 64-bit computing and endianness), building X11-based apps, AquaTerm, and Xgrid. They go into further depth by showing how to link header files (ordinary and precompiled) and libraries in Mac OS X. There are very good examples of building a shared library that contain C functions and dynamically loading libraries.

Part III covers software packages options: Fink, DarwinPorts and creating and installing your own packages. There are step-by-step details on how to install, setup and use both Fink (and FinkCommander) and DarwinPorts, and using Apple's PackageMaker tools.

Using Mac OS X as a server (secure mail, ssh, Apache web, using built-in services through the sharing pref pane, and using the Mac OS X firewall), system management and configuration tools (`top`, `sc_usage`, `vm_stat`, `sysctl`, `nvram`, `scutil`, and third-party apps like Cocktail, MacJanitor and TinkerTool), free databases (SQLite, MySQL, and PostgreSQL), and some Perl and Python extras that are included in Mac OS X are discussed in Part IV.

In Part V, the Appendicxes, cover a basic overview of the Mac OS X GUI and includes a reference to a selection of Mac OS X's Unix development tools. The latter describes the similarities and differences between development tools available on standard Unix systems and Mac OS X Tiger. Also iIncluded are brief descriptions of Xcode tools (`CpMac`, `MvMac`, `Res`), Java (`jar`, `jdb`), text editing and processing (`awk`, `join`, `sed`, `vim`), scripting and shell programming (`echo`, `perl`, `xargs`), file manipulation and sorting (`cat`, `chmod`, `diff`, `rmdir`), and other miscellaneous tools (`apropos`, `passwd`, `su` and `sudo`).

Jepson and Rothman have compiled an excellent book for Unix users who want to come up to speed on Mac OS X quickly and efficiently. **Mac OS X Tiger for Unix Geeks** includes all the basic information that experienced Unix users will need to become comfortable with the Mac platform. The authors cover a wide variety of practical scenarios where Unix sysadmins and developers can put their newfound knowledge to use. If you're comfortable with Unix and want a companion volume to help you unlock Mac OS X's Unix roots, **Mac OS X Tiger for Unix Geeks** is the book for you.

Brian Jepson & Ernest E. Rothman
415 pages
O'Reilly
ISBN: 0596009127
US $34.95

MT

About The Author

Mary Norbury is the IT Director at the Barbara Davis Center for Childhood Diabetes, an affiliate center at the University of Colorado Health Sciences Center in Denver, Colorado. She has too-many-years-to-count experience in cross-platform systems implementation and administration in the education sector. You can reach her at norburym@mac.com.

JobCapture

Automatic Time and Expense Tracking Software

JobCapture is intuitive, user-friendly client-server software that intelligently and automatically tracks working time and activity. Open a document in any application, and it is automatically being timed by JobCapture. This guarantees that *all work activity is recorded*.

- Perfect for ad agencies, design studios, architects, law firms, PR Firms etc.
- Scalable. Flexible.
- Easy to configure.
- Cost effective. Adding additional users is easy and inexpensive.
- Tracks time and expenses.
- Client and Server software is included.
- Easily customized.
- Real-time data delivery.
- Customizable reporting.
- View data numerically or graphically.
- Import/Export data to SQL, mySQL, XML etc.

JobCapture will *increase productivity and your bottom line* by providing accurate records for billing and analysis. *Stop losing money now—Call today* for more information: 973.763.9494. Reps are available 10:00am-5:30pm, EST.

www.captureworks.com

CaptureWorks

Random Cocoa

Implementing Five PRNG Algorithms in Cocoa.

By Jose R.C. Cruz

Introduction

Random numbers are commonplace in the field of software design. They are often used in such applications as cryptography, statistics, and simulations. However, the digital computer is a deterministic machine. It executes specific software instructions in a specific order on data usually provided by the user. Some high-end systems use additional hardware to generate random numbers. These generators use natural events such as radioactive decay or thermal noise as a source for randomness. However, most systems nowadays, use algorithms known as pseudo-random number generators (PRNG).

This article will focus on five commonly known PRNG algorithms. A brief background on each algorithm will be presented, focusing on their respective traits and limitations. Later on, a core Cocoa class will be proposed for implementing each PRNG algorithm. Finally, three test algorithms will be presented. These algorithms are used to determine that a PRNG algorithm is generating a statistically acceptable random sequence.

To further assist software engineers in designing and implementing PRNG Cocoa classes, a demo application known as DiceC is developed to accompany this article. This application implements many of the concepts presented here. Both the application and the XCode project are available for downloading at the MacTech web site http://www.mactech.com.

The Pseudo-Random Number Generator

Basic Concepts

Pseudo-Random Number Generators are algorithms used to generate number sequences that have the appearance of randomness. They achieve this through the use of an iterative mathematical function and/or bit shifting.

As its name implies, PRNGs are not true random number generators. First of all, they all require an initial numeric value known as a ***seed***. The seed value determines the numbers that will be generated in the sequence. Using the same seed will always result in to the same random sequence. This trait, known as ***repeatability***, is desirable in many applications such as cryptography.

Another aspect of PRNG algorithms is that the random sequence repeats itself after *X* number of values. This trait, known as a ***period***, is a distinguishing factor between PRNG algorithms. A well-designed algorithm should have a period as close to (if not greater than) the largest possible random value. A 32-bit PRNG, for example, should be able to generate close to 2^{32} random numbers before the sequence repeats itself.

The Middle Square Method

The Middle Square Method (MSM) is one of the earliest known PRNG algorithms devised for the digital computer. It was proposed by Dr. John von Neumann and used on the ENIAC computer in 1946. It is also a poor generator; its inclusion in this article is mainly for comparative purposes.

Figure 1 shows the generating function used by the MSM algorithm. When generating a new random number, the previous number is first squared. Then only those digits located in the relative middle of the squared result are retrieved as the next random value. The number of significant digits of each random value matches those that comprise the seed value.

To illustrate, if the seed has a value of 1234, first calculate the square of the seed.

```
1234^2 =????????
```

The first random number would then be the middle 4 digits of the squared result.

```
mid(1522756)= 5227
```

To generate the second random number, simply take the first random number and follow the same two steps as shown above.

```
5227^2 = 27321529
mid(27321529) = 3215
```

The MSM algorithm has a number of shortcomings. One is that the algorithm suffers from a very short period of 10^m, where m is the number of significant digits contained by the seed value. In the above example, the seed value only has 4 significant digits. This implies that the algorithm can theoretically generate no more than 9999 unique numbers before it repeats the entire sequence.

Another shortcoming is that the MSM algorithm sometimes implodes into a non-random sequence long before it reaches its maximum period. Unless it is provided with a new seed, the algorithm will start generating a small handful of single number values.

$$R_{n+1} = \text{mid}(R_n^2, m)$$

where:

R_{n+1}	*= new random number*
R_n	*= previous random number*
R_0	*= initial seed value*
mid(,m)	*= m number of digits extracted from the relative middle*

Figure 1. The Middle Square Method.

The Linear Feedback Shift Register

The Linear Feedback Shift Register (LFSR) algorithm uses bit taps and bit shifting to generate a random sequence. As shown in Figure 2, the algorithm first taps specific bits off a theoretical register. It then determines the odd parity of the tapped bits by using an XOR operation. The contents of the register are shifted to the left by one, and the most significantn bit (***msb***) replaced is by the computed parity bit.

The LFSR algorithm is fast, simple and reliable. It has a maximum theoretical period of 2^m, where m is the bit precision of the register. The 32-bit example shown in Figure 2 has a maximum theoretical period of 2^{32}. This implies that up to 4 294 967 values can be generated before the random sequence repeats itself.

Due to its simplicity, the LFSR is often implemented as a hardware generator. It has found popular usage in embedded systems such as spread-spectrum radio and GPS units. Non-linear variants of the LFSR algorithm were also used in cryptography for generating stream ciphers.

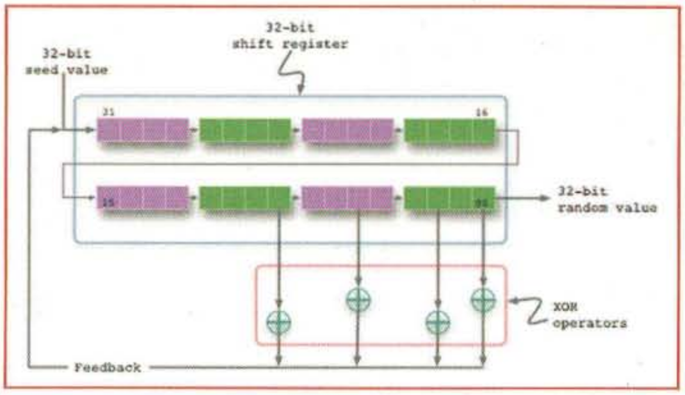


Figure 2. The Linear Feedback Shift Register.

The Linear Congruential Generator

The Linear Congruential Generator (LCG) is one of the oldest and most widespread PRNG algorithms. Many modern operating systems and code libraries use a variation of this algorithm as their random number generator.

Unlike the previous two, the LCG algorithm uses a modular equation (Figure 3) to generate a random sequence. Its overall performance is strongly depended on values chosen for its generator constants **A**, **B** and **M**. A list of common LCG variants, and their respective generator constants, is shown in Table 1.

To ensure an optimal period, the following rules are usually followed when choosing values for the three generator constants:

- ***Constant A is a positive integer much greater than 1.***
- ***Constants B and M needs to be relatively prime with each other. In other words, they share no common factors except 1. This does not imply that either B or M (or both) have to be prime numbers***
- ***The expression (A – 1) should be divisible by all prime factors of M. Consequently, if M happens to a multiple of 4, then (A – 1) should also be a multiple of 4.***
- ***The constant M should be greater than the constants A and B, and the seed value,*** R_0.

To determine if constants **B** and **M** are relatively prime, the algorithm known as ***Euclid's Algorithm*** is used. This algorithm employs successive modular divisions to determine the greatest common denominator (GCD) shared by two integers. If the integers are relatively prime, they will have a GCD of 1. An implementation of Euclid's Algorithm as an ObjC method is shown in Listing 1.

LCG Variant	Constants		
	A	B	M
IBM RANDU	65 539	0	2^{31}
MINSTD	16 807	0	$2^{31}-1$
VAX MTHSRANDOM	69 069	1	2^{32}
BSD rand()	1 103 515 245	12 345	2^{31}
UNIX rand48()	25214903917	11	2^{48}
Cray rand48()	44 485 709 377 909	0	2^{48}

Table 1. List of common LCG variants.

$$R_{n+1} = A \cdot R_n + B \mod M$$

where

R_{n+1} = *new random number*

R_n = *previous random number*

R_o = *initial seed value at $n=0$*

A, B, M = *generator-specific constants (see article)*

Figure 3. The Linear Congruential Generator.

Listing 1. Euclid's Algorithm implemented as an ObjC method.

```
- (BOOL)isValue:(unsigned int)argA
              relativelyPrimeTo:(unsigned int)argB
{
  BOOL chk_flg;
  unsigned int loc_a, loc_b, loc_r;

  // initialize the following locals
  loc_a = (argA > argB) ? argA : argB;
  loc_b = (argA > argB) ? argB : argA;

  // run Euclid's Algorithm
  while (loc_b != 0)
  {
    loc_r = loc_a % loc_b;
    loc_a = loc_b;
    loc_b = loc_r;
  }

  // check the last non-zero remainder
  chk_flg = (loc_a == 1);

  // return the check results
  return (chk_flg);
}
```

The Lagged Fibonacci Generator

The Lagged Fibonacci Generator (LFG) is a specialized version of the LCG algorithm. It uses the Fibonacci sequence of $S_n \cdot S_{n-1}$ to generate its pseudo-random sequence. Unlike other PRNG algorithms, the LFG algorithm uses an initialization vector containing multiple seed values. The vector is often initialized using a different PRNG algorithm. Also, the LFG algorithm uses its random sequence to update its initialization vector, thus improving the overall periodicity of the algorithm.

The generating function of the LFG algorithm is shown in Figure 4. The base modulus, **M**, is usually assigned a value of 2^m, where m is the integer bit size. The token o indicates the binary operator used by the algorithm. This can be any of the following primitives: addition (+), multiplication (x) or exclusive-OR (?). The choice of operator also categorizes the LFG algorithm being used. If the algorithm uses an additive or a multiplicative operator, it is referred to respectively as an ***Additive*** or ***Multiplicative*** LFG. If an XOR operator is used, the algorithm is referred to as a ***Two-Tap Generalized Shift Feedback Register*** (GSFR).

The choice of operator also dictates the theoretical period of the LFG algorithm. If an additive or XOR operator is used, the algorithm has a theoretical period of $(2^k - 1) \cdot 2^{M-1}$. If a multiplicative operator is used, the theoretical period becomes $(2^k - 1) \cdot 2^{M-3}$, which is 1/4 the period of the additive and XOR version.

$$R_{n+1} = R_{n-p} \circ R_{n-q} \mod M$$

where

$\begin{bmatrix} R_o & R_1 & \dots & R_p \end{bmatrix}$ = *initialization vector*

R_{n-p}, R_{n-q} = *last two random values*

$\circ$ = *binary operator (see article)*

M = *base modulus (see article)*

p, q = *generator-specific indices* $p > q > 0$

Figure 4. The Lagged Fibonacci Generator.

The Blum Blum Shub Algorithm

The Blum Blum Shub (BBS) algorithm was developed in 1986 by Lenore and Manuel Blum, together with Michael Shub, as a secure PRNG for cryptographic applications. This algorithm can also be used for simulation projects; however, performance issues render it unattractive for such applications.

The generating function of the BBS algorithm is shown in Figure 5. The generator constant, **M**, is derived as a product of two prime numbers, **p** and **q**. For optimal performance, the following guidelines are proposed when choosing values for **p** and **q**:

- ***Both p and q should be congruent to the modulus*** `3 mod 4`. ***In other words, dividing either p or q by 4 should always result in a remainder of 3.***

- ***The GCD of the expressions*** `(p-1)` ***and*** `(q-1)` ***should be as small as possible to guarantee a large period.***

Unlike the previous PRNGs, only the least significant bits of R_{n+1} are used as the next random value. This ensures that the random sequence is resistant to linear cryptanalysis. For optimal security, it is proposed that no more than `log(log(M))` bits of R_{n+1} should be extracted and used as part of the random sequence.

Multiple Formats.
Multiple Platforms.
Complex Installers.

We have the solution:

StuffIt Engine SDK

Solving the compression & multi-platform puzzle!
www.stuffit.com/sdk/

Licenses start as low as $99/Yr.

Put the power of StuffIt to work for you.

- Adds value to your applications by integrating compression and encryption tools.
- The only tool that supports the StuffIt file format.
- Make self extracting archives for Macintosh or Windows
- Available for Macintosh, Windows, Linux or Solaris

Mac

solaris

StuffIt InstallerMaker

An OS X native version ready for developers!
www.stuffit.com/installermaker/

Prices start at $250

Give your software a solid beginning

- Create Macintosh OS X and Macintosh Classic compatible installers
- Get all the tools you need to install, uninstall or update your software in one complete package
- Add muscle to your installers by customizing your electric registration form to include surveys and special offers.

Allume Systems
www.allume.com
email: dev.sales@allume.com

$$R_{n+1} = R_n^2 \mod M$$

where:

R_0 = *initial seed value*

M = *maximum value based two large prime numbers*

$= p \cdot q$

Figure 5. The Blum Blum Shub algorithm.

The PRNG Cocoa Classes

The Class Structure

The UML diagram of the proposed PRNG Cocoa class structure is shown in Figure 6. The core class, RandBase, provides the code foundation that will be used by 5 other subclasses. Each subclass implements one of the PRNG algorithms covered in this article.

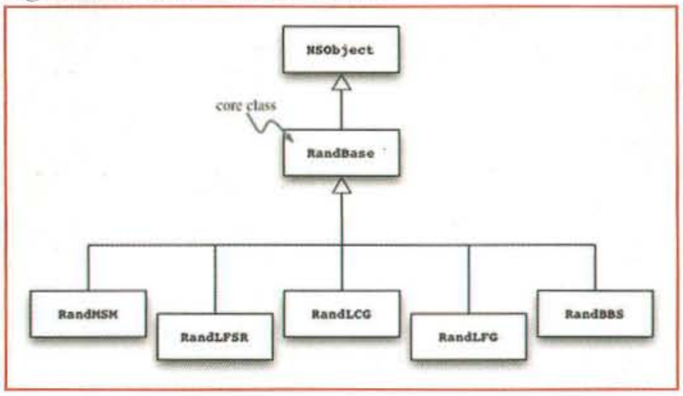


Figure 6. UML diagram of the PRNG class structure.

Figure 7 is the UML diagram of the RandBase core class. For purposes of clarity, I use the more familiar ObjC syntax (as opposed to the standard UML convention) in describing the properties and methods contained by each class.

Three NSString properties store information describing the PRNG class. These are modified using the `setDescription:toString` method. The information stored is then displayed using the `description` method.

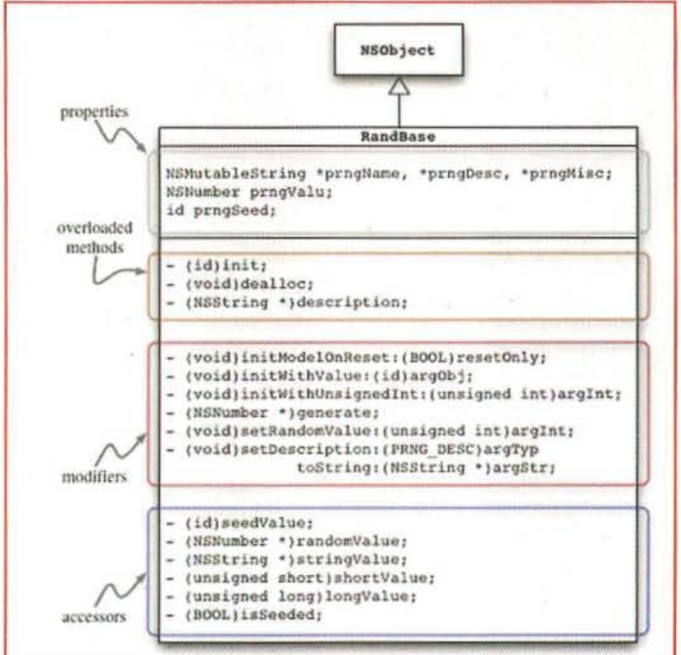
The `prngSeed` property stores the seed value used by the PRNG algorithm. Casting it as a generic datatype, `id`, allows the flexibility to use any Cocoa object as a potential seed. This property is initialized by the core class using either the `initWithValue` or `initWithUnsignedInt` methods.

The `prngValue` property stores the current random value generated by the PRNG algorithm. It is initially set to the same value as `prngSeed` through the placeholder method, `initModelOnReset`. This property is casted as an NSNumber so that the random value can be represented in any one of the following datatypes: string, unsigned integer, double, etc. This property is accessed and modified respectively using the `randomValue` and `setRandomValue` methods.

The placeholder method, `initModelOnReset`, contains the initialization code unique to the PRNG algorithm. Each of the RandBase subclasses overload this method to set their own description strings, default seed values and generator constants.

The placeholder method, `generate`, contains the actual PRNG algorithm. This is also where the actual numeric value of the seed is determined. The RandBase subclasses overload this method with their own implementations. In the core class, this method defaults to the BSD library function, `rand()`, as the random number generator. Also, the generator is initialized in the `initModelOnReset` method using the last random number generated by `rand()`.

Regardless of what algorithm is used, the generate method saves the new random number using the `setRandomValue` method. It also returns the random number as an NSNumber.



```
RandBase
NSMutableString *prngName, *prngDesc, *prngMisc;
NSNumber prngValu;
id prngSeed;

- (id)init;
- (void)dealloc;
- (NSString *)description;

- (void)initModelOnReset:(BOOL)resetOnly;
- (void)initWithValue:(id)argObj;
- (void)initWithUnsignedInt:(unsigned int)argInt;
- (NSNumber *)generate;
- (void)setRandomValue:(unsigned int)argInt;
- (void)setDescription:(PRNG_DESC)argTyp
              toString:(NSString *)argStr;

- (id)seedValue;
- (NSNumber *)randomValue;
- (NSString *)stringValue;
- (unsigned short)shortValue;
- (unsigned long)longValue;
- (BOOL)isSeeded;
```

Figure 7. The RandBase core class.

All of the RandBase subclasses proposed in this article will generate random sequences of 32-bit unsigned integers. Support for other numerical datatypes (64-bit, signed, etc.) can be accomplished by simply modifying each subclass.

The RandMSM Subclass

Figure 8 shows the UML diagram of the RandMSM subclass. This subclass contains the NSRange property, `msm_mask`, which stores the position and length of the

Lasso Professional Server 8.5

Universal Binary for Intel Macs:
Native support for the best speed and compatibility.

Includes AJAX Tags and Functions:
Supports Web 2.0 techniques for modern, dynamic Web design.

Built-in LDAP Integration:
Integrate Lasso with office data and login systems.

Connect to Any Database:
Including Oracle, PostgreSQL, MySQL, FileMaker, ODBC, and JDBC.

Improved DNS, Email, iCal Support:
Extending Lasso's support for Internet protocols and open standards.

4 FREE well commented Lasso solutions to get you started:

Blog: Teaches beginners and advanced users alike how to create an online diary, or web log.

QuickPoll: Add a poll to your Web site with a question and answers, cast a vote or see results.

Collaborate: Enables knowledge sharing among a group of developers and administrators to streamline Web site development from remote locations or geographically dispersed teams.

Message Board: Source code for the Message Board solution is completely open and expandable to suit your needs.

SPECIAL OFFER FOR MACTECH READERS

Try Lasso Professional Server 8.5 FREE for 30 days.

www.TryLasso.com
lassosales@omnipilot.com
800-678-9958

Then get 60 days of FREE support with your purchase.

OMNIPILOT SOFTWARE

Lasso

middle-value mask. The contents of this property are accessed using the `setMask` and `maskValue` methods.

The overloaded method, `generate`, implements the MSM algorithm as shown in Listing 2. This method first retrieves the previous random value and calculates its square product in 64-bit precision. It then converts the resulting 64-bit value to an NSString.

If a middle-value mask has been defined, the method retrieves the mask and applies it to the NSString data. Otherwise, it uses the string length to make a best-guess estimate of the middle value range. The extracted substring is then converted to a 32-bit unsigned integer.

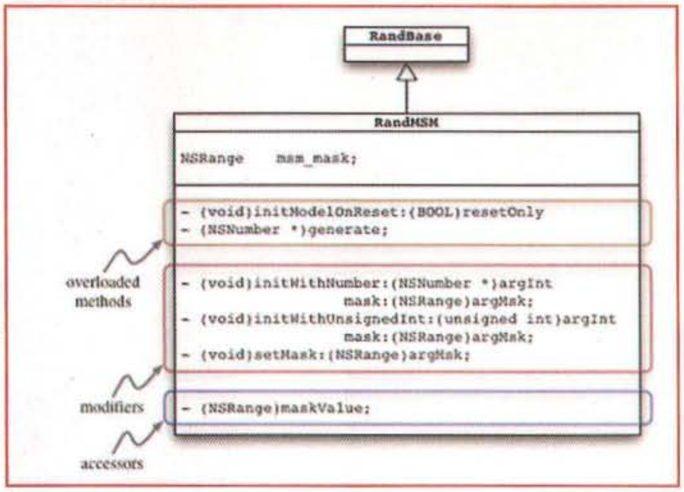

RandBase
RandMSM
NSRange msm_mask;
- (void)initModelOnReset:(BOOL)resetOnly
- (NSNumber *)generate;
- (void)initWithNumber:(NSNumber *)argInt mask:(NSRange)argMsk;
- (void)initWithUnsignedInt:(unsigned int)argInt mask:(NSRange)argMsk;
- (void)setMask:(NSRange)argMsk;
- (NSRange)maskValue;
overloaded methods
modifiers
accessors

Figure 8. The RandMSM Subclass.

Listing 2. The [generate] method (RandMSM.m).

```
- (NSNumber *)generate
{
  NSString *str_val;
  NSRange old_msk, new_msk;
  BOOL    str_chk;
  unsigned int str_pos, str_max, str_len, ui32_val;
  unsigned long long ui64_val;

  // initialize the following locals
  ui64_val = [[super randomValue]
unsignedLongLongValue];
  ui64_val = ui64_val * ui64_val;
  str_val = [NSString stringWithFormat:@"%qu",
ui64_val];
  old_msk = [self maskValue];
  str_max = [str_val length];

  // determine the default MSM mask settings
  str_len = str_max / 2 + (str_max % 2);
  str_pos = str_len / 2;

  // validate the current  mask settings
  if (old_msk.location != old_msk.length)
  {
    // use the current mask settings
    new_msk.location = (old_msk.location > str_max)
      ? str_pos : old_msk.location;
    new_msk.length = (old_msk.length > str_max)
      ? str_len : old_msk.length;
  }
  else
    // use the default mask settings
    new_msk = NSMakeRange(str_pos, str_len);

  // extract the middle value
  str_val = [str_val substringWithRange:new_msk];
  ui32_val = [str_val intValue];

  // save the new random number
  [super setRandomValue:ui32_val];

  // return the new random number
  return ([super randomValue]);
}
```

The RandLFSR Subclass

The UML diagram of the RandLFSR subclass is shown in Figure 9. This subclass contains one private property, `lfsr_taps[]`, which is an array of 32 `BOOL` datatypes. Contents of this property are accessed using the `setTapAtIndex` and `isTapSetAtIndex` methods.

The overloaded method, `generate`, implements the LFSR algorithm (Listing 3). The method first retrieves the previous random value as well as the tap mask. It applies the mask to the random value thus isolating the desired bits. The msb is then determined by computing the odd-parity of the tapped bits. Afterwards, the previous random value is then shifted to the left by one bit, and the msb applied to the shifted result.

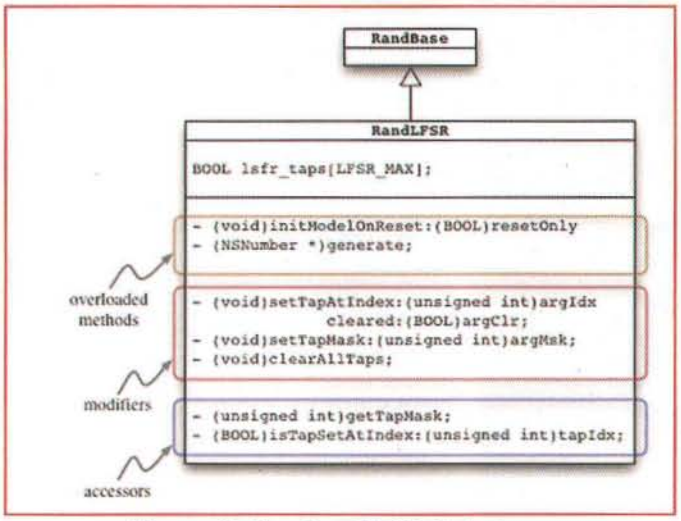
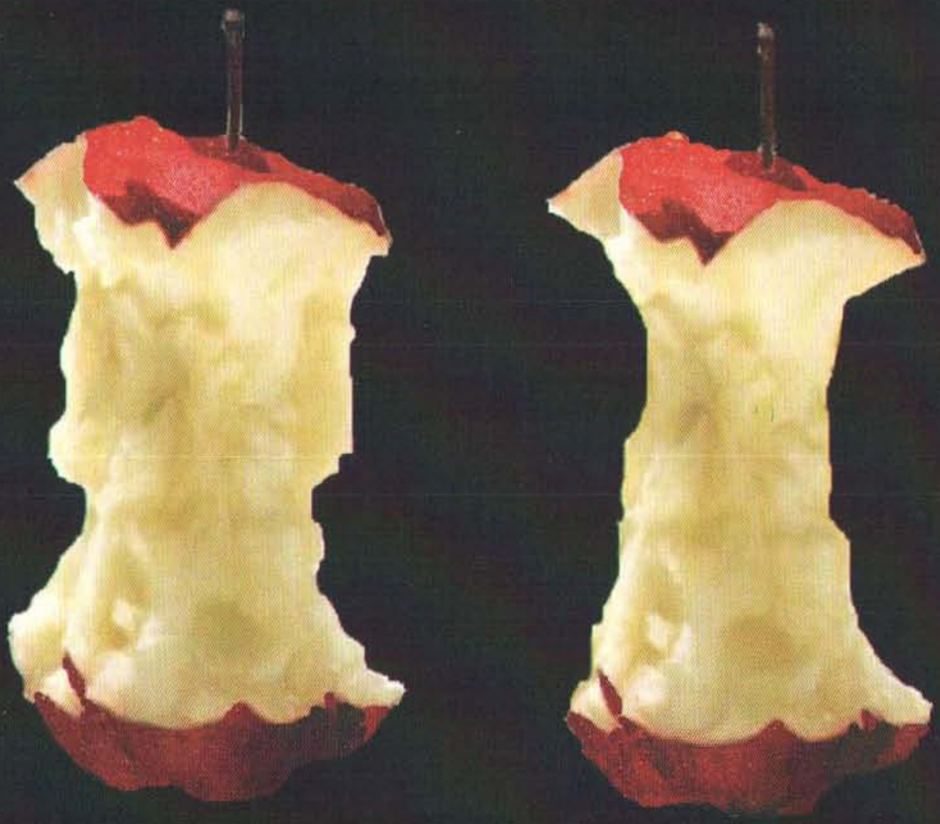

RandBase
RandLFSR
BOOL lsfr_taps[LFSR_MAX];
- (void)initModelOnReset:(BOOL)resetOnly
- (NSNumber *)generate;
- (void)setTapAtIndex:(unsigned int)argIdx cleared:(BOOL)argClr;
- (void)setTapMask:(unsigned int)argMsk;
- (void)clearAllTaps;
- (unsigned int)getTapMask;
- (BOOL)isTapSetAtIndex:(unsigned int)tapIdx;
overloaded methods
modifiers
accessors

Figure 9. The RandLFSR Subclass.

Listing 3. The [generate] method (RandLFSR.m).

```
- (NSNumber *)generate
{
  unsigned int old_val, new_val, msk_val;
  unsigned int bit_pos, bit_val, bit_msb;

  // initialize the following locals
  old_val = [[super randomValue] unsignedIntValue];
  msk_val = [self getTapMask];

  // determine the masked bit values
  msk_val = old_val & msk_val;
   // determine the bit parity [odd-parity of 1]
  bit_msb = 0x0;
```

Really fast compilers are coming ...



Get the most juice out of your new apples

New compilers, debuggers, libraries, and powerful tools are coming soon from Intel® and Absoft for Apple® computers using Intel® processors.

Apply to participate in beta program now!
http://www.absoft.com/trialbeta.html

absoft®

Development tools for Apple® customers since 1984
sales@absoft.com 248-853-0050

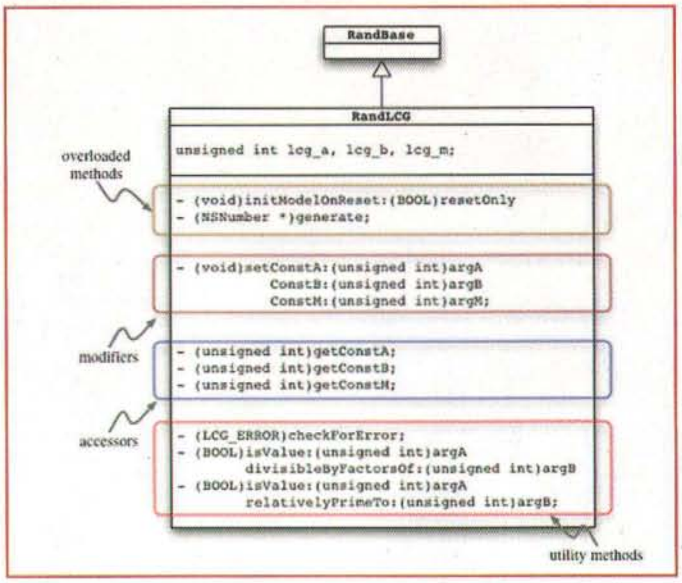
```
for (bit_pos = 0; bit_pos < LFSR_MAX; bit_pos++)
{
  bit_val = 0x1 & msk_val;
  bit_msb = (bit_msb == bit_val) ? 0x0 : 0x1;
  msk_val = msk_val >> 0x1;
}

// update the current random value
new_val = old_val >> 0x1;
bit_msb = bit_msb << (LFSR_MAX - 1);
new_val = new_val | bit_msb;

// save the new random number
[super setRandomValue:new_val];

// return the new random number
return ([super randomValue]);
}
```

The RandLCG Subclass

Figure 10 shows the UML diagram of the RandLCG subclass. This subclass contains three properties (lcg_a, lcg_b and lcg_m) to store the generator-specific constants of the algorithm. The contents of these properties are modified by the setConstA:ConstB:ConstM method, and are accessed by the methods: getConstA, getConstB and getConstM.

The overloaded method, generate, implements the LCG algorithm as shown in Listing 4. The method first retrieves the previous random value as well as the generator-specific constants. It then calculates the LCG generator function shown in Figure 3 at 64-bit precision. The modulus operation, however, is performed at 32-bit precision, resulting into the new random value.

The checkForError function checks to see if the subclass will generate a random sequence with an optimal period. If so, it returns an Err_None; otherwise, it returns the appropriate error flag. The isValue:divisibleByFactorsOf function returns a YES if the specified numbers share the same prime number factors. Finally, the isValue:relativelyPrimeTo function (Listing 1) performs Euclid's Algorithm on the specified numbers. It returns a YES if both numbers are relatively prime. Both functions work in conjunction with the utility function, checkForError.



```
unsigned int lcg_a, lcg_b, lcg_m;

- (void)initModelOnReset:(BOOL)resetOnly
- (NSNumber *)generate;

- (void)setConstA:(unsigned int)argA
          ConstB:(unsigned int)argB
          ConstM:(unsigned int)argM;

- (unsigned int)getConstA;
- (unsigned int)getConstB;
- (unsigned int)getConstM;

- (LCG_ERROR)checkForError;
- (BOOL)isValue:(unsigned int)argA
      divisibleByFactorsOf:(unsigned int)argB
- (BOOL)isValue:(unsigned int)argA
      relativelyPrimeTo:(unsigned int)argB;
```

Figure 10. The RandLCG Subclass.

Listing 4. The [generate] method (RandLCGm).

```
- (NSNumber *)generate
{
  unsigned long long ui64_val, ui64_a, ui64_b;
  unsigned int ui32_val, mod_val;

  // initialize the following locals
  ui64_val = [[super randomValue]
unsignedLongValue];
  ui64_a = [self getConstA];
  ui64_b = [self getConstB];
  mod_val = [self getConstM];

  // calculate the next random value
  ui64_val = ui64_val * ui64_val;
  ui64_val = ui64_val + ui64_b;
  ui32_val = ui64_val % mod_val;

  // save the new random number
  [super setRandomValue:ui32_val];

  // return the new random number
  return ([super randomValue]);
}
```

The RandLFG Subclass

The UML diagram of the RandLFG subclass is shown in Figure 11. This subclass contains two properties (lfg_p and lfg_q) to store the generators-specific constants to be used by the algorithm. It also contains the private property, lfg_ops, which determines the binary operator to be used.

RADTECH™
Solutions that make sense.

OmniCleanz
Cleans what other cleaners leave behind

- 100% Safe – Non-Flammable
- Leaves no Residue – Non-Streaking
- Excellent for LCD and Plasma Displays

10% off OmniCleanz with promo code MT36Q

RadTech.com

no more auction management headaches.

sellitol bestrate 500mg

INDICATIONS: For rapid relief from monthly fees, disorganization, and lost sales. Aids in listing, inventory management, email marketing, trends analysis, customer retention management, and other activities associated with selling profitably on eBay.

DOSAGE: One installation. Subsequent updates are automatic.

DIRECTIONS: List. Sell. Repeat.

ACTIVE INGREDIENTS: automation. scheduled events. custom actions. offline capability. import from turbolister, excel, and databases. ledger. groups. cash flow analysis. support for eBay stores and eBay motors. custom reporting. html editor. bulk listing, ending, revision. relisting. free scheduled listing. offline listing preview. ad templates. image management. watermarks. spell check. consignment management. profiles. vendor management. auto reorder. product sales analysis. questions and feedback management. spam control. fraud management. email templates. invoicing. dispute management. email marketing campaigns.

WARNING: Side effects may include increased sales, higher efficiency, and euphoria.

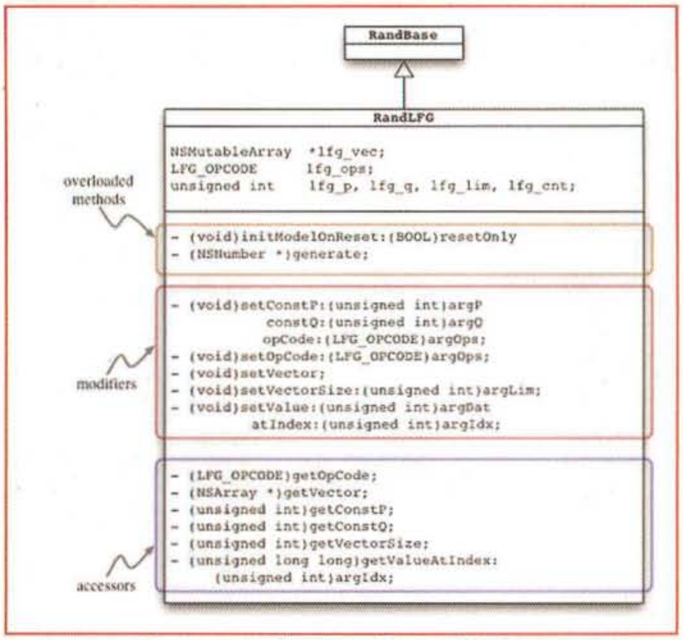
marketblast

please visit www.marketblast.com for a free sample.

The contents of these properties are modified by the `setConstP:constQ:opCode` and are accessed by the methods: `getConstP`, `getConstQ` and `getOpCode`.

The RandLFG subclass also maintains the initialization vector in the form of the NSMutableArray property, `lfg_vec`. Another property, `lfg_lim`, contains the maximum size that the vector is allowed to grow. Lastly, the `lfg_cnt` property, keeps track of the number of random values generated by the subclass. It is used to determine which entry in the initialization vector will be updated.

The overloaded method, `generate`, implements the LFG algorithm (Listing 5). The method first retrieves the previous random value as well as the generators-specific constants. It then determines which seed value to retrieve from the initialization vector. The method calculates the next random value at 64-bit precision, using the selected binary operator. It then updates the initialization vector with the 32-bit result. This is the only subclass that does not save the new random value using the `setRandomValue` method.



```
RandBase

RandLFG
NSMutableArray  *lfg_vec;
LFG_OPCODE      lfg_ops;
unsigned int    lfg_p, lfg_q, lfg_lim, lfg_cnt;

- (void)initModelOnReset:(BOOL)resetOnly
- (NSNumber *)generate;

- (void)setConstP:(unsigned int)argP
           constQ:(unsigned int)argQ
           opCode:(LFG_OPCODE)argOps;
- (void)setOpCode:(LFG_OPCODE)argOps;
- (void)setVector;
- (void)setVectorSize:(unsigned int)argLim;
- (void)setValue:(unsigned int)argDat
         atIndex:(unsigned int)argIdx;

- (LFG_OPCODE)getOpCode;
- (NSArray *)getVector;
- (unsigned int)getConstP;
- (unsigned int)getConstQ;
- (unsigned int)getVectorSize;
- (unsigned long long)getValueAtIndex:
     (unsigned int)argIdx;
```

Figure 11. The RandLFG Subclass.

Listing 5. The [generate] method (RandLFG.m).

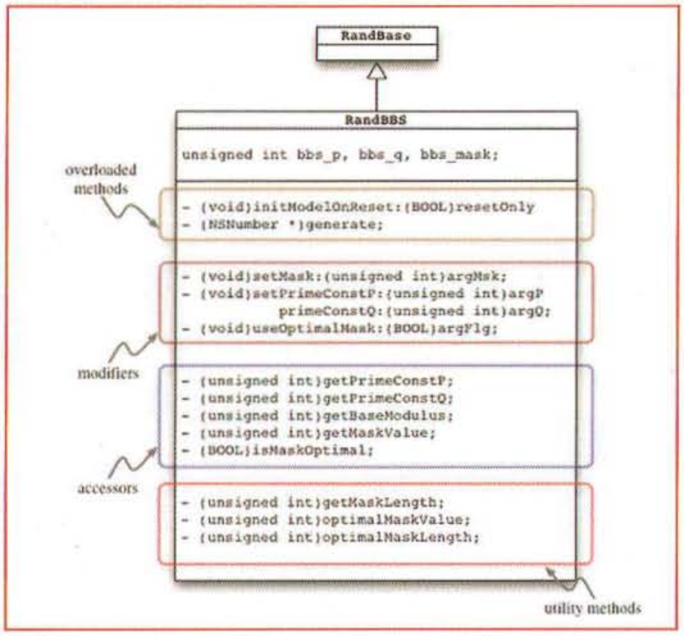
```
- (NSNumber *)generate
{
  unsigned int idx_p, idx_q, vec_lim, ui32_val;
  unsigned long long ui64_p, ui64_q;
  unsigned long long ui64_val, ui64_and, ui64_or;
  LFG_OPCODE arg_ops;

  // validate the initialization vector
  if ([super isSeeded])
  {
    // initialize the following locals
    idx_p = [self getConstP];
    idx_q = [self getConstQ];
    vec_lim = [self getVectorSize];
    arg_ops = [self getOpCode];

    // calculate the vector indices
    idx_p = lfg_cnt - idx_p;
    idx_p = idx_p % vec_lim;
    idx_q = lfg_cnt - idx_q;
    idx_q = idx_q % vec_lim;

    // retrieve the values at the specified indices
    ui64_p = [self getValueAtIndex:idx_p];
    ui64_q = [self getValueAtIndex:idx_q];

    // calculate the next state value
    switch (arg_ops)
    {
      case lfgAdd:
        // — prng:LFG:additive
        ui64_val = ui64_p + ui64_q;
        break;
      case lfgMul:
        // — prng:LFG:multiplicative
        ui64_val = ui64_p * ui64_q;
        break;
      case lfgXOR:
        // — prng:LFG:XOR
        ui64_and = ui64_p & ui64_q;
        ui64_or = ui64_p | ui64_q;
        ui64_val = ui64_or & ~(ui64_and);
        break;
    }

    // include the modulus base
    ui32_val = (unsigned int)(ui64_val %
DFLT_BASE);

    // update the initialization vector
    lfg_cnt++;
    idx_p = lfg_cnt % vec_lim;
    [self setValue:ui32_val atIndex:lfg_cnt];

    // save the new random number
    [super setRandomValue:ui32_val];

    // return the new random number
    return ([super randomValue]);
  }
  else
    // return the current seed value
    return ([super seedValue]);
}
```

The RandBBS Subclass

Figure 12 shows the UML diagram of the RandBBS subclass. This subclass contains two properties (`bbs_p` and `bbs_q`) to store the prime constants, **p** and **q**. It uses the property, `bbs_m`, to store the generator constant **M**. The subclass also has the property, `bbs_mask`, which is the user-defined bit mask used to extract the new random value. Finally, it has the `BOOL` property, `bbs_opt`, dictating whether to use the user-defined bit-mask or the optimal one discussed earlier in this article. All of these properties are accessed by their respective methods.

The overloaded method, `generate`, implements the BBS algorithm as shown in Listing 7. The method first retrieves the previous random value and the base modulus, **M**. It calculates the square of the random value at 64-bit precision, and then determines the modulus result at 32-bit precision. Afterwards, the method applies the appropriate bit mask to the modulus result. It then repeats all of the aforementioned steps until a full 32-bit random value is formed.

By default, the subclass uses the optimal bit mask to extract the bits that will comprise the new random value. The methods, `optimalMaskLength` and `optimalMaskValue` (Listing 6), are used to determine the optimal bit mask. However, the subclass can also use a user-defined bit mask. This is done by setting the bit mask using the `setMask` method and passing a `NO` to the `useOptimalMask` method.



```
unsigned int bbs_p, bbs_q, bbs_mask;

- (void)initModelOnReset:(BOOL)resetOnly
- (NSNumber *)generate;

- (void)setMask:(unsigned int)argMsk;
- (void)setPrimeConstP:(unsigned int)argP
          primeConstQ:(unsigned int)argQ;
- (void)useOptimalMask:(BOOL)argFlg;

- (unsigned int)getPrimeConstP;
- (unsigned int)getPrimeConstQ;
- (unsigned int)getBaseModulus;
- (unsigned int)getMaskValue;
- (BOOL)isMaskOptimal;

- (unsigned int)getMaskLength;
- (unsigned int)optimalMaskValue;
- (unsigned int)optimalMaskLength;
```

Figure 12. UML Diagram of the RandBBS Subclass.

Listing 6. Calculating the optimal bit mask (RandBBS.m).

```
// — Calculate the number of bits in the optimal mask
- (unsigned int)optimalMaskLength
{
  unsigned int ui32_len;
  double dbl_val;

  // calculate the optimal mask length
  dbl_val = (double) [self getBaseModulus];
  dbl_val = log(dbl_val);
  dbl_val = log(dbl_val);
  ui32_len = ceil(dbl_val);

  // return the calculation results
  return(ui32_len);
}

// — Calculate the optimal BBS mask value
- (unsigned int)optimalMaskValue
{
  unsigned int ui32_len, ui32_msk, ui32_bit;

  // calculate the optimal mask value
  ui32_len = [self optimalMaskLength];
  ui32_msk = 0x0;
  for (ui32_bit = 0; ui32_bit < ui32_len;
ui32_bit++)
  {
    ui32_msk = ui32_msk | 0x1;
    ui32_msk = ui32_msk << 0x1;
  }
  ui32_msk = ui32_msk >> 0x1;

  // return the calculation results
  return (ui32_msk);
}
```

Listing 7. The [generate] method (RandBBS.m).

```
- (NSNumber *)generate
{
  unsigned long long ui64_val;
  unsigned int ui32_val, ui32_mod;
  unsigned int ui32_msk, msk_len, msk_cnt, msk_max;

  // initialize the following locals
  ui64_val = [[super randomValue]
unsignedLongValue];
  ui32_mod = [self getBaseModulus];

  // is an optimal mask requested?
  if ([self isMaskOptimal])
  {
    msk_len = [self optimalMaskLength];
    ui32_msk = [self optimalMaskValue];
  }
  else
  {
    msk_len = [self getMaskLength];
    ui32_msk = [self getMaskValue];
  }

  // calculate the new random value
  msk_max = 32 / msk_len;
  ui32_val = 0x00;
  for (msk_cnt = 1; msk_cnt <= msk_max; msk_cnt++)
  {
    // calculate the next random value
    ui64_val = ui64_val * ui64_val;
    ui64_val = ui64_val % ui32_mod;

    ui32_val = ui32_val | (ui64_val & ui32_msk);
    if (msk_cnt < msk_max)
      ui32_val = ui32_val << msk_len;
  }

  // save the new random number
  [super setRandomValue:ui32_val];

  // return the new random number
  return ([super randomValue]);
}
```

Testing the PRNG Subclasses

A variety of algorithms are available for testing the efficacy of PRNG implementations. These algorithms examine the generated random sequence to determine whether a PRNG is properly configured. They also determine whether or not the generated sequence is statistically suitable for the task at hand.

PRNG test algorithms range from the elegantly simple Lempel-Ziv to the mathematically oriented maximum-of-t test. They all require a large number of random values in order to be effective. A large sample size helps reveal indications of poor randomness such as skewing ness. In fact, it is quite common to have a PRNG generate at least 1000 random values for testing purposes.

This article introduces three simple and effective test algorithms: Lempel-Ziv, Run Method, and Spectral

Method. The demo application, DiceC, uses the Spectral Method as its test algorithm.

The Lempel-Ziv Method

The Lempel-Ziv method is one of the simplest test algorithms for a PRNG. In its most basic form, the PRNG in question first outputs its random sequence into a binary file. This file is then compressed using the LZ algorithm or any of its variants. Some implementations do away with file I/O by compressing the random sequence directly in the memory.

Since the LZ algorithm works by identifying and consolidating byte patterns, a pure random sequence should have a near-zero compression ratio. The compression ratio of PRNG random sequence, however, should be very low if it has a near optimal period. Higher compression ratios are often good indications of non-random patterns present in the sequence.

The Lempel-Ziv method is an effective baseline test when comparing various PRNG algorithms. It is also useful for fine-tuning algorithms such as LCG or LFG, whose overall performance is strongly depended on its generator constants or seed values.

The Run Method

The Run method converts a random sequence into a series of states. An ***ascending*** state is where each subsequent random value increases with respect to time. It is represented by a (+) sign. The opposite would be a ***descending*** state, which is represented by a (-) sign. If the values remain unchanged, the state is considered to be ***neutral*** and is represented by a (0).

To illustrate, the random sequence

```
255 758 029 974 984 320  583 765
```

has the following random values exhibit an ascending state:

```
255 -» 758   029 -» 974   974 -» 984
320 -» 583   583 -» 765
```

whereas the following exhibit a descending state.

```
758 -» 029   984 -» 320
```

This reduces the random sequence to the following state sequence

```
+ - + + - + +
```

Once the state sequence is determined, it becomes a simple matter of examining the sequence for any patterns. Long successions of one or more states are undesirable as they indicate a numerical bias in the PRNG random sequence.

The Spectral Method

The Spectral Method divides a random sequence into specific intervals (sometimes known as a ***spectral period***) and then plots the number values within each interval as a scatter graph. The result is a spectrum of data points that can then be examined for patterns.

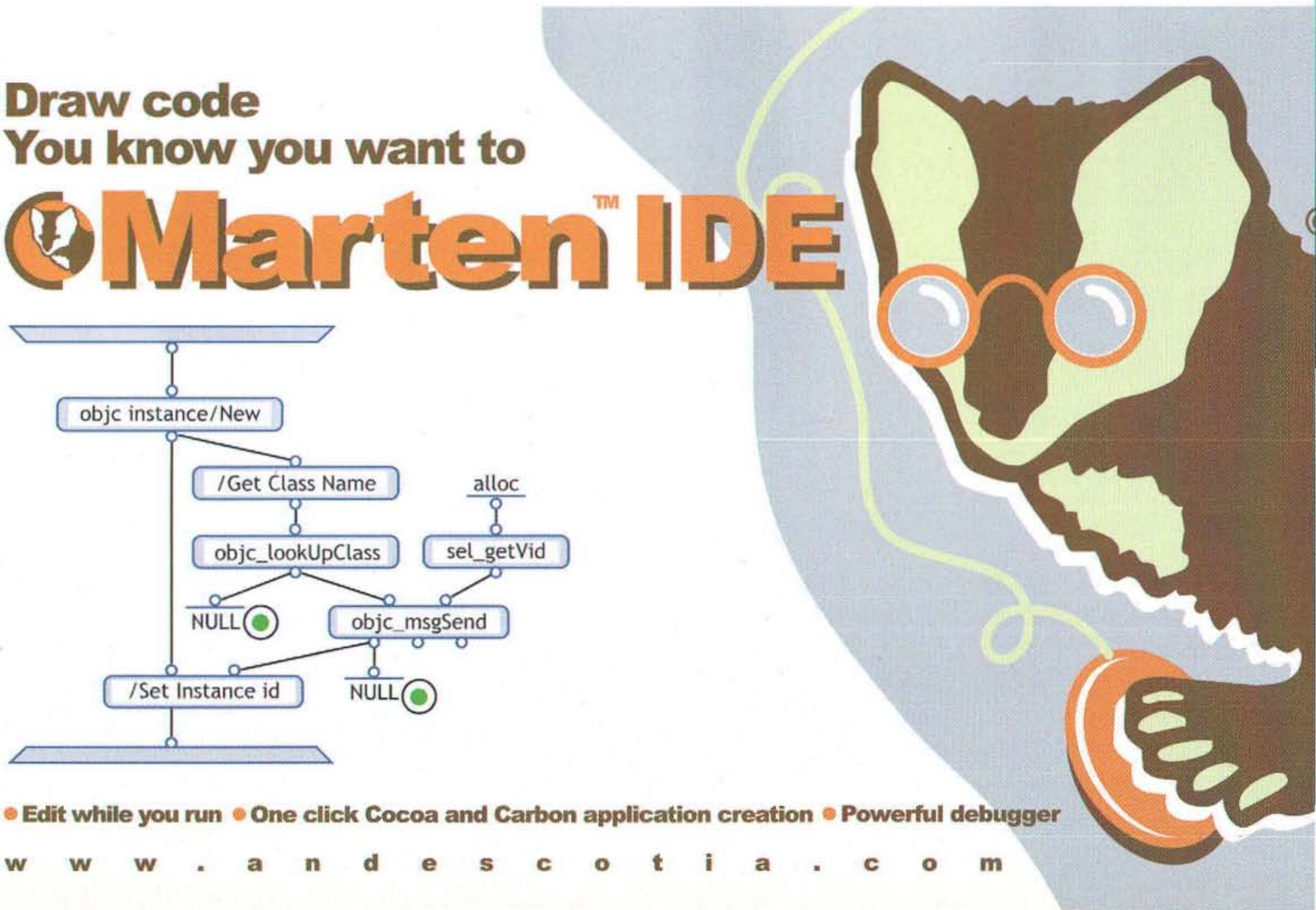


Figure 13 shows a typical spectrum graph for the BSD `rand()` function. The random sequence consists of 1000 32-bit unsigned integers. It is generated using a seed value of `0x1ebbccf`. The scatter graph is then drawn using a spectral period of 200 values.

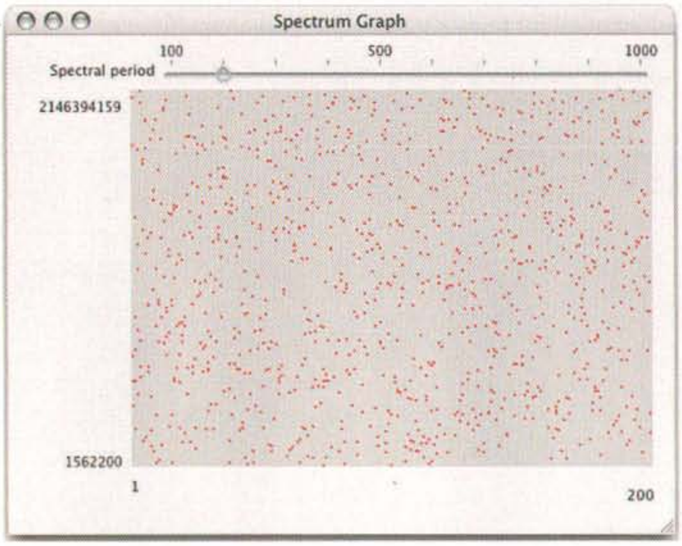


Figure 13. Spectral graph of the BSD rand() function.

Notice that the data points are well distributed across the resulting graph. There are no areas where the data points would cluster together forming a pattern. Now, compare this with the spectrum graph for the RandMSM subclass(Figure 14). The random sequence of 1000 integers used in this graph is generated with a seed value of `0x457`. Notice the 6 dashed lines located at the bottom of the graph. This indicates that the sequence converged to 6 distinct values after 11 iterations. This is an undesirable trait, especially considering that the implemented MSM algorithm is expected to have a period of 9999 values.

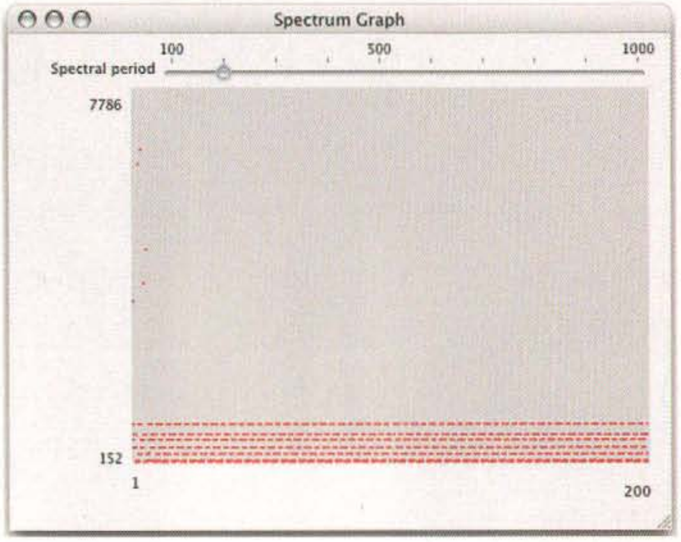


Figure 14. Spectral graph of the MSM generator.

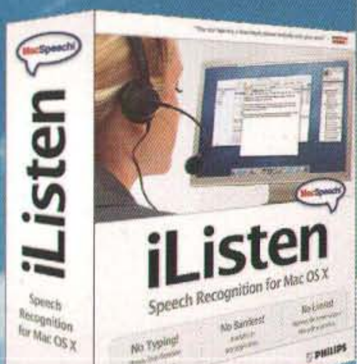
Peace for your fingers

Speech Recognition for Mac OS X

iListen, the top-selling Speech Recognition system for Mac OS X, lets you do almost anything with your voice. Our natural speech engine and voice command & control system takes the pain out of typing.

Optional unique ScriptPaks let you select all your menus by voice, there's even a ScriptPak for BBEdit, the developers choice of text and source editing.

For a limited time you can get a free BBEdit ScriptPak with any purchase of iListen from our online store. The BBEdit ScriptPak allows you to do almost anything in BBEdit with your voice that you would normally do with your keyboard and mouse. To get the free ScriptPak you must use the coupon code FREE-BBEDIT when checking out.



See www.macspeech.com/promo/bbedit.html or call (603) 251-1477 for more details.

MacSpeech

www.macspeech.com

The spectrum graph for the other three subclasses fared much better by comparison. Shown in Figure 15 is the spectrum graph for the RandLFSR subclass. Its random sequence is generated using a seed value of `0xbeef`. Like that of the BSD `rand()` function, the graph shows well-distributed data points with no noticeable patterns.

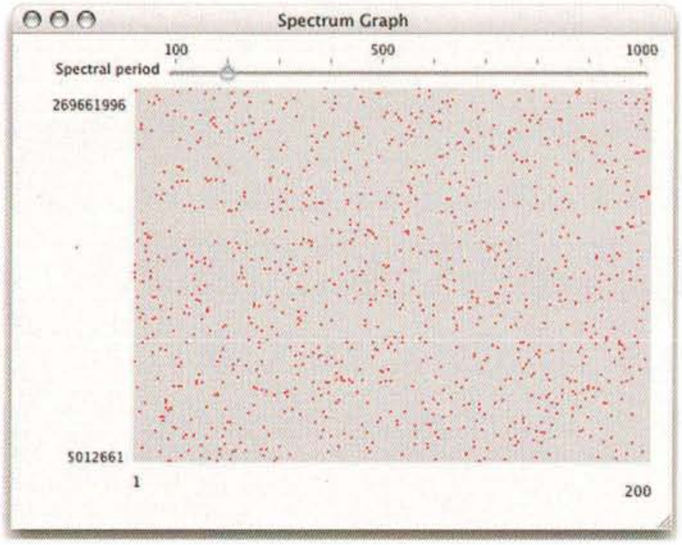


Figure 15. Spectral graph of the LFSR generator.

Summary

The ability to generate random sequences is an essential component of any software code library. Since digital computers are deterministic machines, the only feasible way of generating random sequences is through the use of a pseudo-random number generators or PRNGs. Five different PRNG algorithms are then introduced and discussed.

Later, the class, RandCore, is proposed to serve as the basis for developing a PRNG Cocoa object. Five different subclasses are then based off the RandCore class, with each subclass implementing one of the five PRNG algorithms. Both RandCore and its five subclasses are implemented in the demo application, DiceC. Also, introduced are three test algorithms for testing the PRNG subclasses. One of these algorithms, the Spectral Method, is implemented in DiceC to evaluate the performance of RandCore and its five subclasses.

Bibliography and References

Harvey Gould, Jan Tobochnik, Wolfgang Christian. ***Random Number Sequences***. An Introduction to Computer Simulation Methods. pp.245-249. PDF document posted on 2005 May 19. (c) 2005 Gould, Tobochnik and Christian.

Makoto Matsumoto, Takuji Nishimura. ***Mersenne Twister:A 623-dimensionally equidistributed uniform pseudorandom number generator***. ACM Transactions on Modeling and Computer Simulations:Special Issue on Uniform Random Number Generation. 1998.

Karl Entacher, Hannes Leeb. ***Inversive Pseudo-random Number Generators:Empirical Results***. Parallel Numerics 95. Department of Mathematics. University of Salzburg, Austria. 1995 Sep 27-29.

Jagannath Pisharath. ***Linear Congruential Number Generators***. Newer Math. 2003 Sep.

Martin Geisler, Mikkel Krøigård and Andreas Danielsen. ***About Random Bits***. 2004 Dec 03.

Wikipedia. Euclid's Algorithm. In Wikipedia, the free encyclopedia. The Wikipedia Community, Oct 2005. Online:
http://en.wikipedia.org/wiki/Euclidean_algorithm.

Wikipedia. Pseudorandom number generator. In Wikipedia, the free encyclopedia. The Wikipedia Community, Oct 2005. Online: http://en.wikipedia.org/wiki/PRNG.

About The Author

JC is a freelance engineering consultant currently residing in North Vancouver, BC. He divides his time between custom application development for OS X, technical writing, and teaching origami to children at the local district libraries.

Rumpus FTP

The Complete Internet File Transfer Server For Macintosh

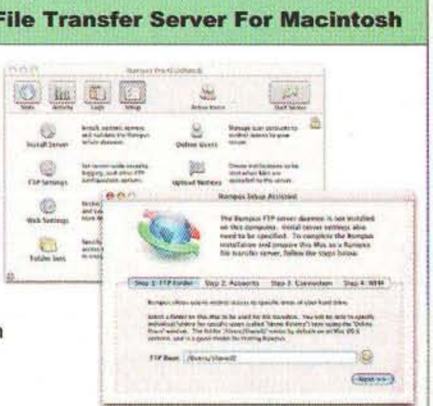
Turn any Mac into a file transfer server...

Simple installation & setup

Dozens of advanced features

Fast, efficient server

Local or remote administration

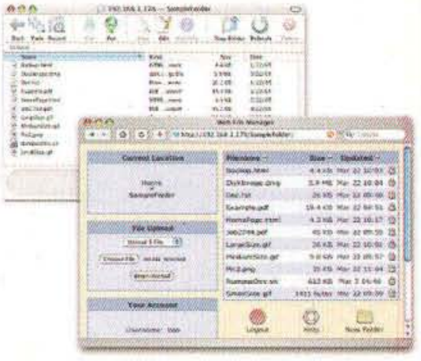


To let anyone send or receive files...

Transfer files with any Web browser

Compatible with all major FTP clients

Support clients on Mac or PC



Download A Free, Fully-Functional Trial Copy Today

http://www.maxum.com/

AppleScript's Variable Types and You

What you don't know about AppleScript's variable types can hurt you (or slow down your scripts!)

By Ryan Wilcox

Introduction

Those who know AppleScript well are very familiar with obscure AppleScript tricks, magical incantations to make their scripts run faster. These incantations are usually due to an understanding of AppleScript's data sharing mechanisms, which are thoroughly explained in this article.

Learning how different types of variables work is an important step for any AppleScripter. Sometimes AppleScript behaves in ways that are unintuitive and often frustrating to those who don't understand what's going on. To those that don't understand these different types, some of AppleScript's behavior, looks like bugs in the language, but this is rarely the case.

This article discusses three classifications of AppleScript's variable types, examines methods of appending an item to a list, and then examines the different kinds of copy operations available to a scripter. This article is not meant to be an introduction to the language, but rather read by someone who has a working knowledge of AppleScript.

Variable Types

Pure, or "Vanilla" AppleScript (AppleScript that does not depend on any third party extensions aka: OSAXen, nor applications) has three different classifications of variable types: scalar types, container types, and other types. The first classification, scalars, is a type that can only hold one item. An example being a variable that contains the number 27. Numbers, reals, as well as dates, strings, and texts are also scalars.

The second classification is container types. A container type is a type that can hold multiple values. For example, a scalar could contain the number `27`, while a list could contain `{27, 3, 5, 6}`, and a record could contain `{item1: 27, item2: 3, item3: 5, item4: 6}`. Lists, records and script objects are container types.

The last classification is other types. Other types include references, as well as constants and unit types.

The information presented in this article (and this section in particular) is all backed up by experimentation, most of which was condensed into a test script. This script contains nearly four-dozen handlers that test the assertions made in this article. Most of the snippets of code you'll find are from the test script as well. The full script is found on the Web at: http://www.wilcoxd.com/ articles/asreferences/ Each snippet of code in this article is backed by a handler in the testing framework, which will return `true` if the conditions have been met, or `false` if the test has failed. Handlers can easily be located by using the Table Of Contents at the top of the test script. Doing the timed tests included in the test script requires the Jon's Commands OSAX, found at http://www.seanet.com/~jonpugh/ The script can be run without the timed tests by setting the `doTimings` property to `false`.

Scalar and Container Types Compared

There is a difference in how AppleScript deals with scalars and how it handles container types. Container types, internally, use something the

AppleScript Language Guide calls "Data Sharing" (page 206). The understanding of this "data sharing" concept first requires an understanding of how variables work in AppleScript.

An AppleScript record, deconstructed, might be the easiest way to see the behavior of AppleScript's variable types, especially when accompanied by some illustrations.

```
set myRec to {varOne: 1, varTwo: 2, varThree: 3}
```

Figure 1 represents what `myRec` looks like:

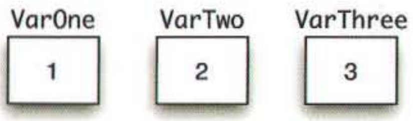


Figure 1: myRec with three scalar items inside it

Now we create a new item in `myRec`, `illustrated in Figure 2:`

```
set myRec to myRec & {varFour: 4}
```

and represent it by Figure 2:

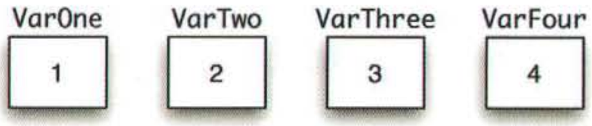


Figure 2: myRec with four scalar items inside it

Simple enough. Figure 3 shows, the situation changes when we set `varOne`, `varTwo`, and `varThree` to be container types instead:

```
set myRec to { varOne: {1}, varTwo: {2}, varThree:
{3} }
```

`myRec` represented graphically looks like Figure 3:

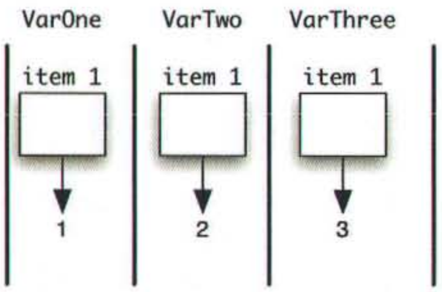


Figure 3: Diagram of our variables as container types

As seen in the diagram, container types refer to their contents by pointing to where the data is "over there". In essence, container types in AppleScript work a lot like aliases in the Finder. An alias points to another file or folder, allowing you to access that item from a different logical location in your file structure. Likewise, container types let you refer to the same value "from" multiple variables. Aliases also take up less hard disk space than making a duplicate of the item itself, an advantage shared by AppleScript's container types: the same values only exist in one place in memory, instead of having multiple copies of the same data.

This means we can change two things with list items, just as we can with aliases. We can change the contents of the variable (the data that is shared), or we can change the variable itself (where it "points" to). First, Figure 4 shows `myRec` when we add another element to it:

```
set myRec to myRec & {varFour: varThree of myRec}
```

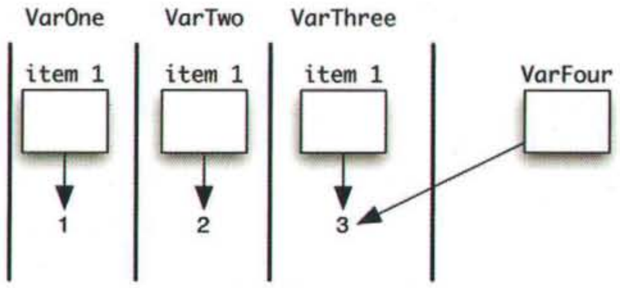


Figure 4: varFour shares the contents of varThree

Notice how `varFour` shares `varThree`'s data. As you might expect, any change to {3} will be mirrored in both `varThree` and `varFour`, as the data is shared between these two variables, as seen in the following line of AppleScript (also shown in Figure 5).

```
set item 1 of varThree of myRec to 4
```

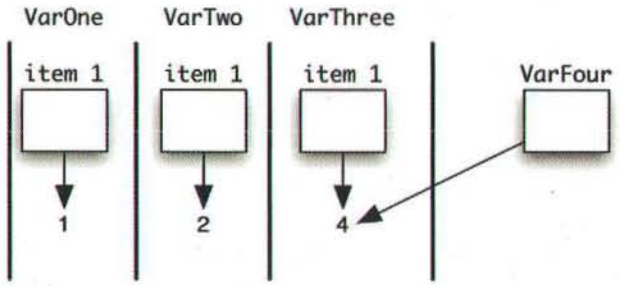


Figure 5: Changing the contents of varThree

Changing where a variable points is also possible, shown in Figure 6:

```
set varThree of myRec to {2.5}
```

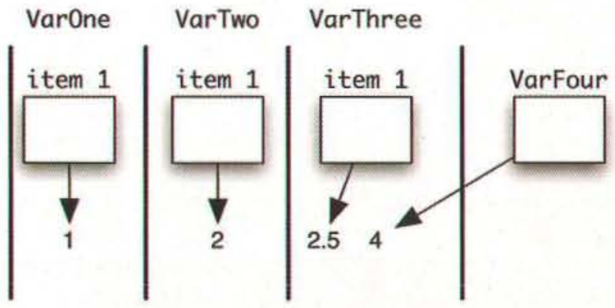


Figure 6: Changing varThree

The above line changes only where `varThree` points to, as opposed to the contents (aka: the shared value) of what it was sharing with `varFour`. It would be as if you created two aliases in the Finder (alias A and alias B), both of them pointing to the same document, and then you selected a new original for alias B. Alias A continues to point to the original item; only alias B points somewhere else.

AppleScript shares the contents of container items, instead of sharing the item itself. This can be proven with a handler from the test script, `testContainerItemsShareScalars()`.

```
to testContainerItemsShareScalars()
  set myItem to {1}
  set myList to myItem
  set myList to myList & myItem
  set item 1 of myItem to 45

  return (myList is equal to {1,1} and myItem is
equal to {45})
end testContainerItemsShareScalars
```

If item 1 and item 2 of `myList` shared item 1 of `myItem` itself, they would reflect the change in line 5. Instead, item 1 and item 2 share the ***contents*** of item 1 of myItem, just like our example in Figure 6. Items in `myList` remain unchanged: they still share the same value, even if `myItem` does not anymore. The test script has two tests that further explore this functionality: `testContainerItemChangeReflectedInShared()`, and `testContainerItemChangeReflectedInShared()`. `testContainerItemChangeReflectedInShared()` proves that even if item 1 of myItem is a container, instead of a scalar, the behavior is the same as `testContainerItemsShareScalars()`. `testContainerItemChangeReflectedInShared()` proves that if item 1 of `myItem` is a container, that changing the contents of `myItem` is reflected in `myList`.

C/C++ or Java programmers might recognize that container items act a lot like pointers or references in their languages. They would be correct, to a point. This article pointedly avoids the term "reference" to refer to container types, to make it more accessible to scripters not familiar with those languages, and to avoid confusion with AppleScript's actual reference type, discussed in the next section. Care should be taken, especially when reading AppleScript mailing lists, as container items are often called "references", despite not being associated with AppleScript's actual reference type. AppleScript's container types are different from the reference type because they are automatic, and don't have any of the restrictions of actual references. The test script tries to avoid using the term "reference" as a substitute for "container types", but sometimes uses it for brevity.

The Reference Type

In vanilla AppleScript, the reference type exists to allow the scripter to point to scalar data. Try running the following handler:

```
on run
  set x to true
  set y to a reference to x
  set contents of y to false
  log "x = " & x as string
  log "y = " & y as string
end run
```

```
(*Event Log Output:
  x = false
  y = false
*)
```

Here `y` is a reference that shares data with `x`. Remember that the difference between scalar types and container types is that container types point to data "over there". The reference type points to data "over there" as well, and reflects changes to the shared data just like container items do.

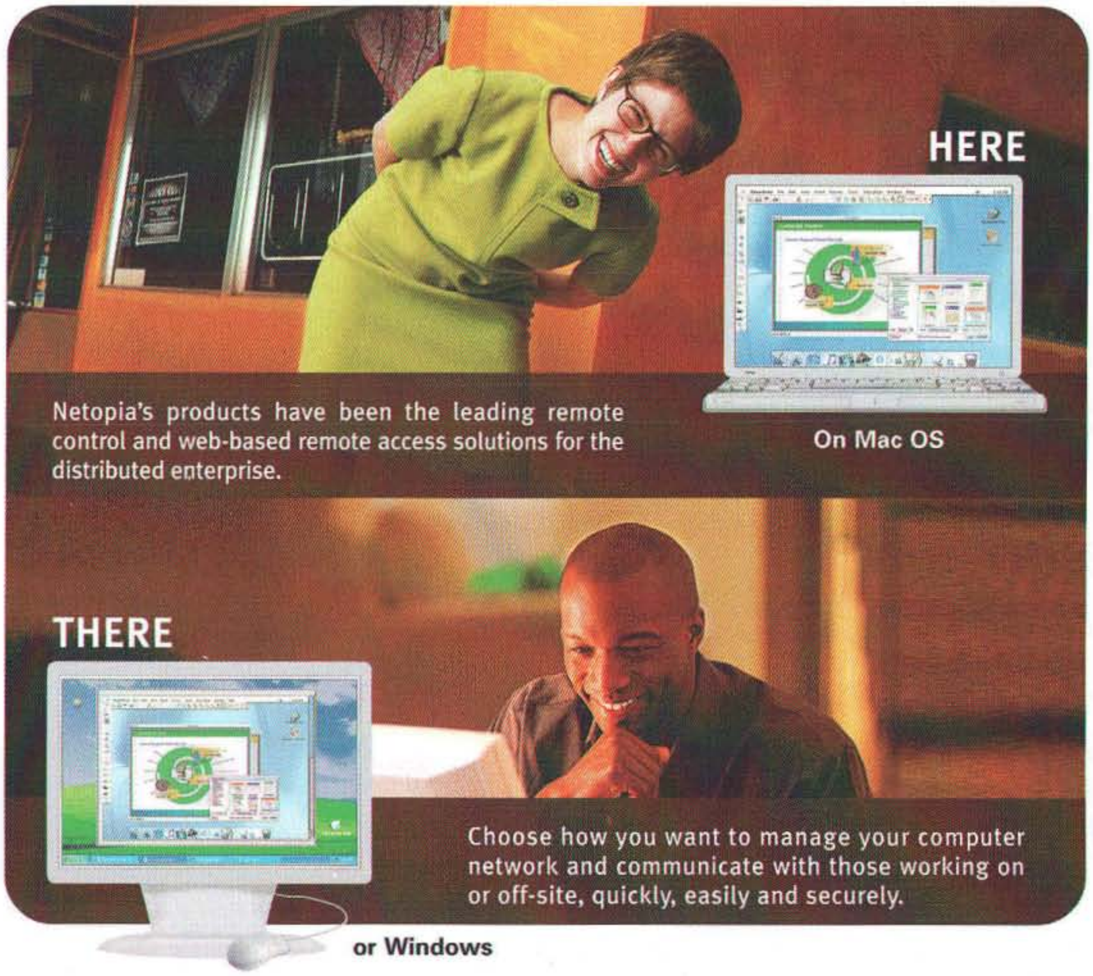
Below is a snippet of code, based on the `changeAReferenceType()` handler from the test script:

```
set refProp to "world"
set y to a reference to refProp
set z to a reference to refProp

set y to "hello"
set contents of z to "foo"

—refProp is equal to "foo"
—and (y is equal to "hello"
—and (contents of z is equal to "foo"))
```

First it's important to note that `refProp` is actually a property. We'll get into why later in this section, but for now just remember that fact if you're trying to reproduce this test at home.

Lines 1-3 of the handler create our three variables, pictured in Figure 7:

Stay In Control Wherever You Go.



Netopia's products have been the leading remote control and web-based remote access solutions for the distributed enterprise.

Choose how you want to manage your computer network and communicate with those working on or off-site, quickly, easily and securely.

Timbuktu® Pro

Whether you're at home or at work, using a Mac OS or Windows platform, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them. Transfer files and folders quickly and easily, and communicate by instant message, text chat, or voice intercom.

eCare

eCare's web-based remote access capabilities enable any user to securely place their virtual eyes and fingers directly on the remote client's desktop for collaboration, problem resolution, and improved customer satisfaction. Using only a web browser, eCare's secure and fast screen sharing, file transfer, live chat, URL push and live session recording make it the ideal solution for the conference room, classroom, or help desk – live, on demand.

Download an evaluation or buy it online: www.netopia.com
Call us at 1-800-485-5741

timbuktu™ pro · eCare netopia.

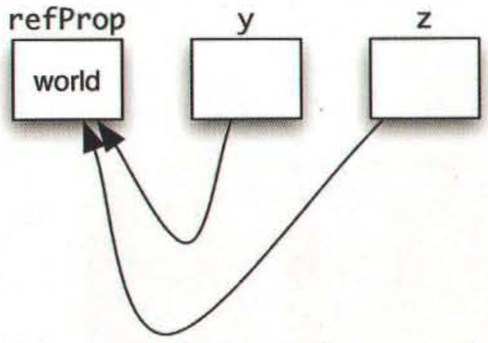


Figure 7: Initial state of refProp, y and z

Line 4 changes the value of y to "hello". Remember that container types have two different types of values: what they point *to*, and the contents of the variable. In Figure 7, y points to `refProp`, and points at the value "world". Reference types work the exact same way. Figure 8 shows the state of y after line 4.

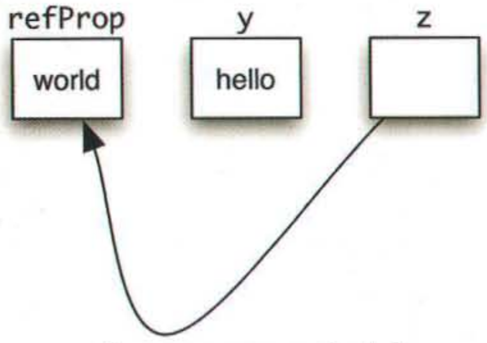


Figure 8: set y to "hello"

Here we have only changed what y points to, so z has the same value as it had before.

Now, in line 5, we change the value z points to, using AppleScript's `contents of` syntax. `Contents of` allows us to change the value being pointed to by the reference type.

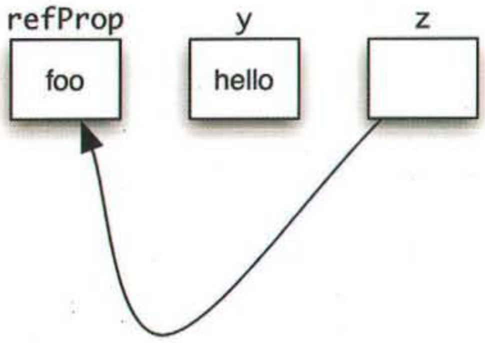


Figure 9: set contents of z to "foo"

Here we are changing the value z points to, the value of `refProp`.

Reference types seem like they would be useful, but they have some pretty limiting drawbacks, according to ***AppleScript: The Definitive Guide*** (Section 11.5), and our tests here. Try running the following script:

```
to main()
  set x to true

  set y to a reference to x
  set the contents of y to false
  —AS runtime error:"can't make 'x' into type reference
end main

main()
```

While y points to x, we get an error when we try to get the contents of the reference. ***AppleScript: The Definitive Guide*** has a work-around: make x a property. This is what `changeAReferenceType()` does with `refProp`.

```
property x : true

on main()
   set y to a reference to x
   set the contents of y to false
   log "x = " & x
   log "y = " & y
end main

main()

(*Event Log Output:
  x = false
  y = false
*)
```

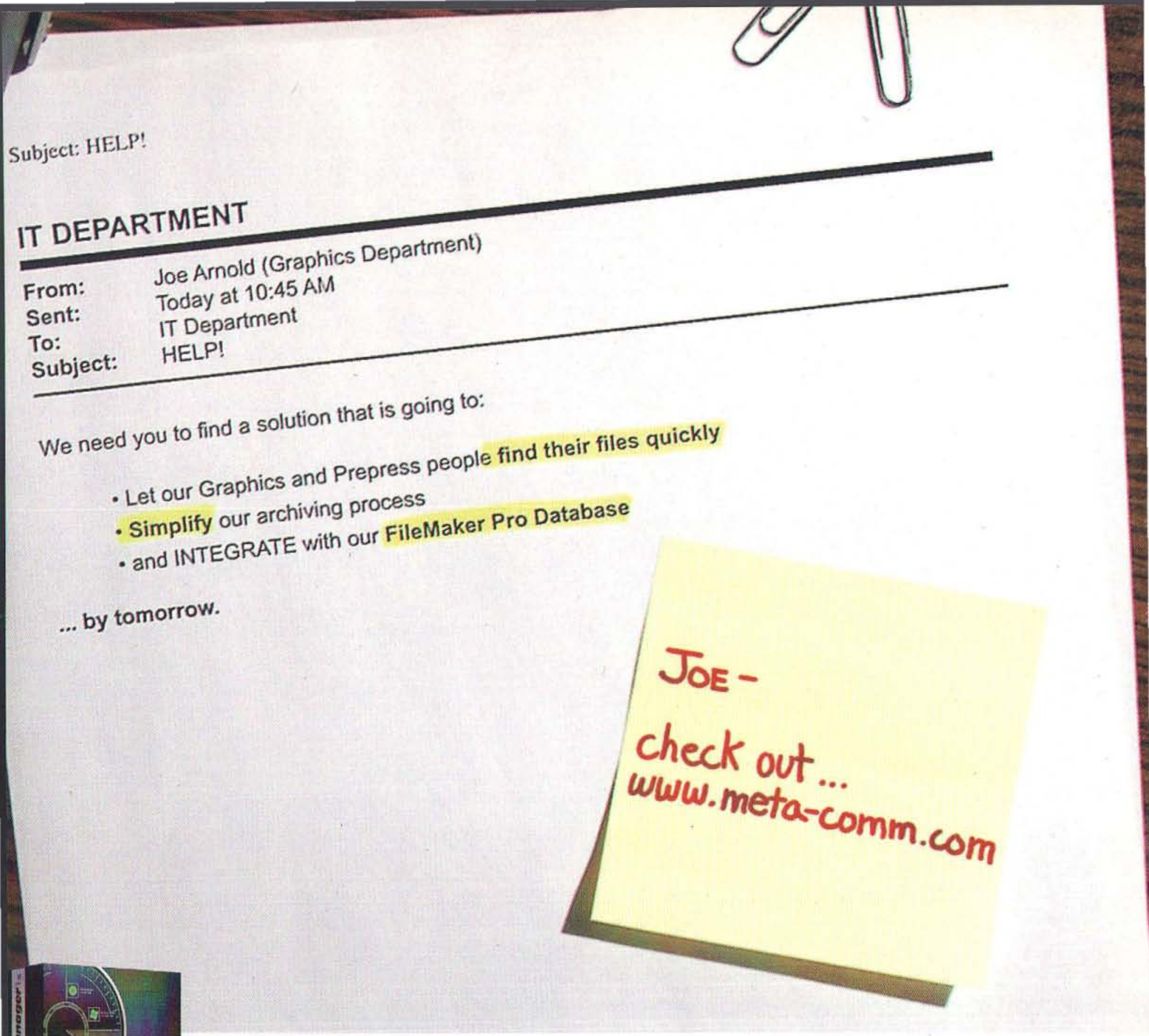
Further experimentation leads to an interesting fact: you ***can*** create a useful reference type using a `reference to` ***if and only if*** you are in the (implicit or explicit) run handler, as the following script proves.

```
on run
   set x to true
   set y to a reference to x
   set contents of y to false
   log "x = " & x as string
   log "y = " & y as string
end run

(*Event Log Output:
  x = false
  y = false
*)
```

For those of you not aware, when running a script AppleScript first looks for a handler explicitly named `run()`to execute, and if it doesn't find it, then executes any commands not in a handler, assuming that (the implicit run handler).

Referencing a list in the run handler:

```
on run()
   set x to [true, " foo"]
   set y to (a reference to x)
   set contents of y to [false, " bar"]
   log "x = " & x as string
   log "y = " & y as string
end run

(*Event Log Output:
  x = false bar
  y = false bar
*)
```



Subject: HELP!

IT DEPARTMENT

From: Joe Arnold (Graphics Department)
Sent: Today at 10:45 AM
To: IT Department
Subject: HELP!

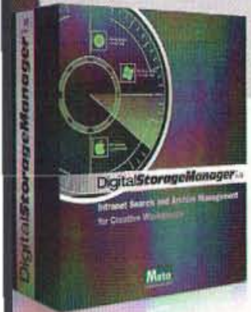
We need you to find a solution that is going to:

- Let our Graphics and Prepress people find their files quickly
- Simplify our archiving process
- and INTEGRATE with our FileMaker Pro Database

... by tomorrow.

JOE -

check out...
www.meta-comm.com



New Version

DigitalStorageManager™ 1.5

Intranet Search and Archive Management for Creative Workgroups

Digital Storage Manager is the first intranet search and archive management system for creative and prepress workgroups. Digital Storage Manager offers:

- Ability to search and instantly find files by production related metadata.
- A simple one-step Archiving solution including integration with: CA Software ARCserve, Rimage, EMC Dantz Retrospect, Mac OS X File Burner, Roxio Toast, Microsoft Removable Storage Manager.
- Secure Web Portal provides your customers, freelancers or other departments the ability to search and download files via the web.
- Integrates with existing job or production management systems, including FileMaker Pro.

Meta COMMUNICATIONS

Download a 21 Day Trial or Call to Schedule a Demo

Web: www.meta-comm.com Tel: 1-800-771-6382 Email: sales@meta-comm.com

The reference type seems like a useful way to save memory (and time) when you want one variable to mirror a scalar variable's value. In reality, thanks to the restrictions of the reference type, it is probably better and easier to use a container type instead.

Beyond our explorations of the reference type, the different mechanisms of copying can confuse scripters and could lead to slower than expected performance in scripts. The next section of this article covers this important topic.

Conclusions on Variable Types

Knowing the difference between scalar and container types is an important step in understanding AppleScript 's behavior in certain circumstances (like using the `set` command). AppleScript's reference type, while not as useful or powerful as other types, is important to know about and may prove useful in your script writing.

The next section shows several test handlers that further illustrate the difference between scalar and container types. Now, test your knowledge by trying to guess the output of the event log of the following handlers, based on the information in the previous section.

Test Your Knowledge!

This section is composed of 5 functions from this article's test script. See if you can guess what the output of the event log will be in each case.

The Test

```
on xAnYScalars()
  set x to true
  set y to x
  set y to false
  log "x = " & x
  log "y = " & y
end xAnYScalars

on xAndYList()
  set x to {true}
  set y to x
  set item 1 of y to false
  log "x = " & x as string
  log "y = " & y as string
end xAndYList

on xAndYListCopy()
  set x to true
  copy x to y
  set y to false
  log "x = " & x as string
  log "y = " & y as string
end xAndYListCopy

on copyWithLists()
  set x to {true}
  copy x to y
  set item 1 of y to false
  log "x = " & x as string
  log "y = " & y as string
end copyWithLists
```

The Answers

```
xAnYScalars() Event Log
  (*Event Log Output:
      x = true
      y = false
  *)

xAnYList() Event Log
  (*Event Log Output:
      x = false
      y = false
  *)

xAndYListCopy() Event Log
  (*Event Log Output:
      x = true
      y = false
  *)

copyWithLists() Event Log
  (*Event Log Output:
      x = true
      y = false
  *)
```

The Answers, Explained

These tests show the basics of how AppleScript's variables react differently to different situations. First, in `xAnYScalars()`, we have a test that sets two scalars. The line `set y to x` copies the value from x and puts it into y. Since y is a copy, we change its contents and not have it reflected in x.

`xAndYList()` is different in that it deals with container types. Using `set` with a container type only points to the original data. Thus, in `xAndYList()`, y shares x's data, claiming it as its own, and changing the value of any element inside of y will be reflected in x.

You might expect the same results from `xAndYListCopy()` as those from `xAndYList()`. No, for this function uses the `copy` keyword, which results in a duplication of all the data, down to every last scalar value, from x into y. This duplication (also called "deep copy") means that there are two different copies of {true}, and changing one copy does not affect the other. This would be like duplicating a file in the Finder, and making a modification to one, copies contents, one file contains the new data, while the other file remains the same as it was before. Copy operations are covered later in this article.

The next section will take the knowledge of scalars and container types, and apply it to the simple operation of adding an item to the end of a list.

Appending To A List Under the Microscope

Appending an item to a list happens a fair deal in AppleScripts, and can be a source of slowness if not done correctly. In the following sections we'll investigate several different ways to append an item to a list, discussing both speed and differences in the operations performed. Space considerations prevent detailed analysis with all permutations, but we'll cover the more interesting ones in this article. Other append constructs can be found in the test script.

By downloading the test script you can see the timings of the tests yourself, on your machine, and also further examine the handlers this author used to gather the data presented in this series of articles

The benchmark machine for these tests was a (admittedly ancient) 400Mhz Powerbook G4 running OS X 10.4.3 with 768 MB RAM. Timings are given in ticks, which is 1/60th of a second.

copy myList & myListItem to myList

copy myList & myListItem to myList copies both myList and myListItem - changes made to either variable don't get reflected in the new list. twiddleCopyListItemList() shows us trying to change the contents of item 1 of myList, which is a duplicate of (instead of sharing data with) myItem.

```
to twiddleCopyListItemList()
  set a to 1
  set myList to {{a}, {a}, {a}, {a}, {a}}
  copy myList to deepCopy
  set shallowCopy to items of myList
  set item 1 of item 4 of deepCopy to 42 - not
reflected
  set item 1 of item 5 of myList to 23 - reflected
  return (myList is equal to {{1}, {1}, {1}, {1},
{23}}) and (deepCopy is equal to {{1}, {1}, {1},
{42}, {1}}) and (shallowCopy is equal to {{1}, {1},
{1}, {1}, {23}})
end twiddleCopyListItemList
```

This function will be revisited later in the article, but note how the change to item 4 of deepCopy does not change the value of shallowCopy or myList. We used copy to construct deepCopy, making a duplicate of the data. Also notice how changing value 5 of myList is reflected by shallowCopy.

Now, the timing of this construction: timeCopyListAndListToList() took a minimum of 0 ticks, max of 3, and averaged 1.11 ticks on the benchmark machine.

set myList to myList & myListItem

The next way to append an item to a list is to use set myList to myList &... There are things to be aware of while appending container items to a container. The first is that, appending two lists together in this method will result in a "flat" list. Take the following handler, from the test script:

```
to setListToListAndContainerListItemAppend()
  set a to {1, 3}
  set b to {2}
  set b to a & b
  return b is equal to {1, 3, 2}
end setListToListAndContainerListItemAppend
```

b grew to accommodate all of a, creating a new item for every item of a. b is now 3 items long (count of a + count of b = 3.)

One must be careful when initially creating myList, to avoid inadvertently changing values you didn't expect, as shown in the following handler:

```
to testEmptyListsChangesContents()
  set emptyList to {}
  set myItem to {1}
  set myList to emptyList & myItem
  set item 1 of myList to 123
  set myList to myList & myItem

  return (myItem is equal to {123} and myList is
equal to {123, 123})
end testEmptyListsChangesContents
```

One might have expected myList to be {1, 123}, instead of {123, 123}. On line 3 of this handler, myList behaves as if we had used the "set myList to myItem" syntax, where myList simply reflects myItem, instead of having item 1 of myList share myItem. Figure 10 shows this distinction.

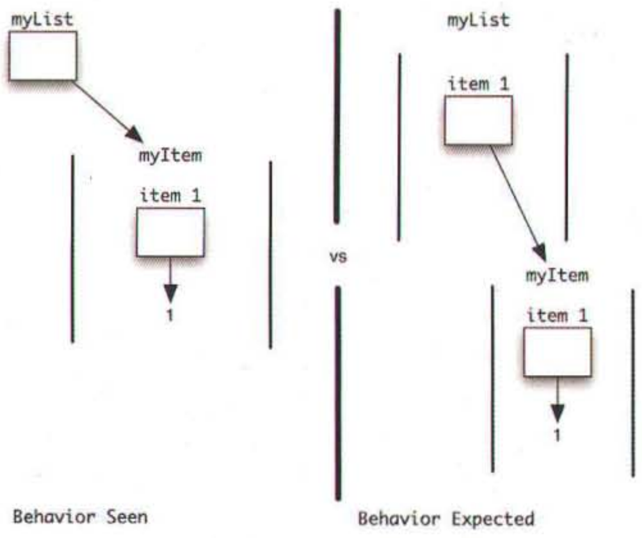


Figure 10: Seen vs expected behavior in testEmptyListsChangesContents()

timeSetListToListAndList(), in the test script, gave us a min value of 0 ticks, a max of 2, and an average of .8 ticks. This is very close to the timing for set myList to myList & myListItem, timed with timeSetListToListAndScalar() (found in the test script).

set end of myList to myListItem

AppleScript's syntax has many nice features to work on lists. first of, end of, rest of, and some item of, to name a few. testFirstOfReassignRefItem() in the testing script shows, for example, that first of will replace the first item of the list (instead of creating a new first item, such that the previous first item is now the second item, and so on)

The `end of` syntax appends an item onto a list. For an example, examine the handler we use to time this construct:

```
to testSimpleEndOfWorks()
  set a to {2}
  set b to {1}
  set end of b to a
  return b is equal to {1, {2}}
end testSimpleEndOfWorks
```

Notice that item 2 of b is {2}, and not simply 2, as in a `set myList to myList & anotherList` construction, it is nestled, instead of flat.

A word to the wise with this construction: do not append a variable to itself using this technique. At best it will result in a stack overflow when you go to display it, at worse it could cause an infinite loop. Even if the list is sharing another variable and you try adding that variable onto the list. `simpleAddSelfToEndOfSelfErr()` of the test script shows trying to append an item onto itself, and the handler `addSelfToEndLogOverflowErr()` shows the more exotic case.

Another difference with the `end of` syntax vs. the `set... to...` syntax is that changes are shared. The handlers `setEndOfSharedData()` and `addToOneNotShared()` in the test script prove this, but they can be distilled with two simple examples:

```
set a to {1}
set b to a
set b to a & 45
a
® {1}
to
set a to {1}
set b to a
set end of b to 45
a
® {1, 45}
```

On line 3 of the first construct, it is like b is created all over again, which would explain the discrepancy in speed between this and `set end of`. It also explains why, in the second snippet, a contains {1, 45} while in the first example, a remains {1}. In other words: the second example acts on b itself, and since b shares data with a, logging a at the end of the second script will show it identical to b, because b is just a reflection of a.

Being able to refer to the end of the list has some timing advantages. The benchmark machine took a minimum of 0 ticks to complete `timeSetEndOfListToList()`, 1 tick max, and an average of .09 ticks. Compare this to `set myList to myList & myListItem`, and you'll see it was almost 10 times faster!

Conclusions on Appending To Lists

By far the fastest method of appending an item to a list is to use the `set end of...` to construction. The `timeSetEnd OfList100ItemsList()` and `timeSetEndOfList100 ItemsScalar()` handlers, presented in the test script, append 100 lists and scalars with 100 items, respectively, to a list, constructing a 1000 item list. On the benchmark machine, these tests were identical (min of 0, max of 1, average of 0.13). The test script shows timed examples of `copy myList & myItem to myList constructing this kind of list`, one constructing the 100 item list using the `set end of...` syntax, and one using `copy myList & myItem to...` (`timeCopyScalarAndListToList 100Items()` and `timeCopyList AndListToList 100Items()` respectively). These two functions take an average of 2.5 ticks per run on the benchmark machine, proving that careful use of AppleScript can speed up your scripts, even with something as simple as appending items to lists.

The next section discusses the different methods to copy variables in AppleScript. Along with appending to a list, copying variables draws on much of our previous knowledge about AppleScript's variable types.

The Right Copy For The Right Situation

AppleScript has two different types of copy: a shallow copy and a deep copy. Early in this article, we mentioned AppleScript's `copy` construction. A deep copy, which is triggered in AppleScript using the `copy` keyword, will perform a deep copy: duplicating all of the data in a variable, even if you have scalars nested inside of lists inside of records inside another list. A deep copy of such an object would be very expensive because of all the data involved. AppleScript would duplicate each and every item in this complex structure.

An alternative would be a shallow copy. Like a deep copy, a shallow copy duplicates all of the items in a list. Unlike a deep copy, any container items in the original list will create a new item in the destination. This item will share the same data as the original item. Our scalar deeply nested inside a list would be very fast with a shallow copy, because AppleScript would share the original data and put that in our new list. To expand on our earlier example of aliases, shallow copying a list simply creates another alias "pointing to" the same file as the first.

Let's examine these two kinds of copy individually:

Shallow Copy

A shallow copy creates items that share data with their respective items in the original list. This means that when you shallow copy, data is not duplicated, but the items of the new list share their data with the items of the original list.

When you create a shallow copy, items added or removed from either list are not reflected in the other list, but changing the contents of an item will be reflected.

You can shallow copy a list in AppleScript by using the `items of` syntax.

```
set shallowCopy to items of myList
```

It is important to reiterate here: the items of `shallowCopy` point to the data shared by `myList` directly, they are not depending on `myList` for anything beyond the initial setup. We can change an item on `myList` to point somewhere else, yet `shallowCopy` remains the same.

```
to shallowCopyChangeOne()
  set a to {1}
  set b to {2}

  set myList to {a, b}
  set shallowCopy to items of myList

  set item 1 of myList to 0
    return (myList is equal to {0, {2}}) and (a
is equal to {1}) and (shallowCopy is equal to
{{1}, {2}})
end shallowCopyChangeOne
```

Figure 11 shows what `shallowCopy` looks like when we create it on line 4 of `shallowCopyChangeOne()`.

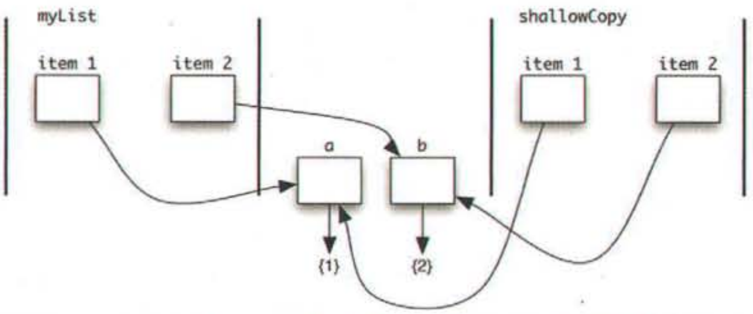


Figure 11: shallowCopy as created.

Notice how both items of the lists share the same data, but are two separate sets of items. You can really see this in line 5, where we set item 1 of `myList` to 0.

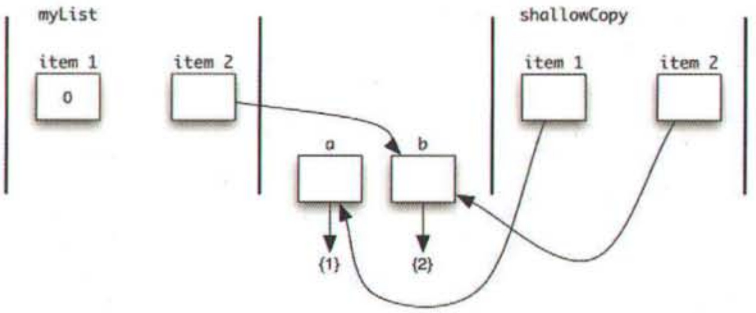


Figure 12: set item 1 of myList to 0

Connect • Communicate • Collaborate • Securely

Because meetings are more productive when people are on the same page... of their calendars

Get the message

Kerio MailServer

A groupware alternative to Exchange that syncs calendars, contacts and email with Entourage and Outlook. Integrated anti-spam and McAfee virus filtering provide secure, junk-free email for users on any platform.

Mac OS X · Linux · Windows

www.kerio.com

McAfee Proven Security

KERIO

Line 5 is setting what item 1 of `myList` points to. Since `myList` and `shallowCopy` are separate entities, and we did not change the data shared between them (a), our change does not affect anything outside of `myList`.

Unlike `shallowCopyChangeOne()`, where we changed what item 1 of `myList` shares, `shallowCopy ChangeSharedValue()` changes the contents of an item shared between the two lists. Watch and see:

```
to shallowCopyChangeSharedValue()
  set a to {1}
  set b to {2}

  set myList to {a, b}
  set shallowCopy to items of myList
  copy myList to deepCopy

  set item 1 of item 1 of myList to 0

  return (myList is equal to {{0}, {2}}) and
(shallowCopy is equal to {{0}, {2}}) and (deepCopy
is equal to {{1}, {2}})
end shallowCopyChangeSharedValue
```

Line 6 changes the value of (item 1) of a, which is shared by both `myList` and `shallowCopy`. Since `shallowCopy` and `myList` both share the contents of a, the change is shown in both lists.

Shallow copy only has an advantage when dealing with container items. Just like `set`, scalar items are duplicated, so there is no speed or memory gain for using a shallow copy on a list of scalar values.

Deep Copy

A deep copy is when AppleScript duplicates all of the data inside a container type. This means that both lists duplicate each other's data, and modifying even the contents of one will not affect the other, because there is no sharing going on at any level. You can deep copy a list by using the `copy to` syntax:

```
copy myList to deepCopy
```

Since `deepCopy` is a copy of the values in `myList`, we can change one list without affecting the other in any way.

```
to deepCopyChangeOne()
  set a to {1}
  set b to {2}

  set myList to {a, b}
  copy myList to deepCopy
  set item 1 of a to 42

  return (myList is equal to {{42}, {2}}) and (a is
equal to {42}) and (deepCopy is equal to {{1},
{2}})
end deepCopyChangeOne
```

Line 4 of this handler duplicates `myList`, bit for bit, creating `deepCopy`. Figure 13 illustrates what `myList` and `deepCopy` look like on line 4. Line 5 of this handler changes item 1 of a. A's data is shared by `myList`, but not by `deepCopy`, thus the changes are not reflected.

air power

No surgery required!

Works for all models & speeds

Introducing
The Bird Killer

Signal. Strong signal. That's what you'll get from this inventive new wi-fi extending product from QuickerTek. The Bird Killer attaches to the outside of any iBook or Powerbook — no internal tinkering needed! To learn more about this, and other Mac-friendly inventions, visit ***www.quickertek.com***

Also available...

Antennas
Whips | Plug n' Plays
Base Station

Transceivers
Base Station | Mac Mini
Power Mac

Handles
PowerBook | iBook
TiBook

Wireless Performance Products *for Mac by*
QUICKERTEK
www.quickertek.com ***GO!***

Our headsets brought the world closer to the moon.
Now they can keep you close to your music.

PULSAR™ 590A

ULTIMATE STEREO BLUETOOTH® HEADSET

MUSIC AND MOBILE - Listen to music or movies, and answer phone calls.
TALK AND LISTEN LONGER - Up to 12 hours talk time and 10 hours stereo listening.
UNIVERSAL ADAPTABILITY - Enjoy Bluetooth functionality with most devices.

In 1969, a Plantronics headset carried those first words from the moon,"That's one small step for man..." Our latest Bluetooth headset, the Plantronics Pulsar 590A, was designed for versatility here on Earth. Seamlessly switch between your Bluetooth phone and your favorite music, so you'll never miss a call. And with the Universal Adapter, you can experience Bluetooth stereo listening on most laptops, Macs, TVs, DVD players and MP3 players. So your next mission will be as enjoyable as ever.

We've been to space, now it's your turn. Enter to Win a Trip to Space—visit Plantronics.com or text "headset" to SPACE (77223).

PLANTRONICS.
SOUND INNOVATION™

Bluetooth®

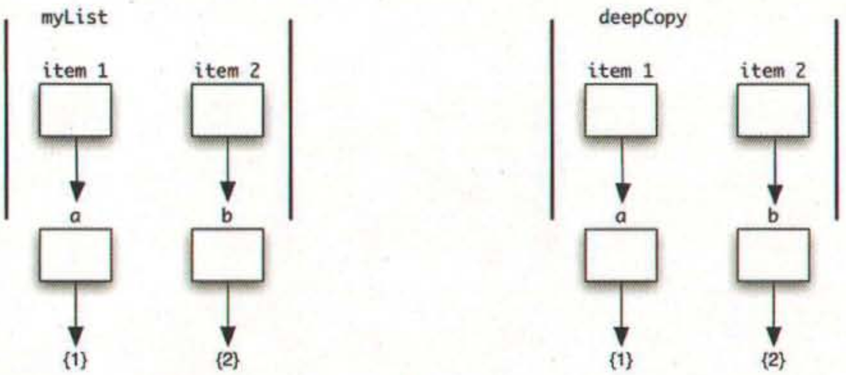


Figure 13: myList vs deepCopy

Our test framework's `twiddleCopy ListItemList()` shows that one can change a scalar value in one list and not have it reflected in the deep copied list. twiddleCopyListItemList() is a more complex example of the same topics shown by `deepCopyChangeOne()`

```
to twiddleCopyListItemList()
  set a to 1
  set myList to {{a}, {a}, {a}, {a}, {a}}
  copy myList to deepCopy
  set shallowCopy to items of myList
  set item 1 of item 4 of deepCopy to 42
  set item 1 of item 5 of myList to 23

  return (myList is equal to {{1}, {1}, {1}, {1},
{23}}) and (deepCopy is equal to {{1}, {1}, {1},
{42}, {1}}) and (shallowCopy is equal to {{1}, {1},
{1}, {1}, {23}})
end twiddleCopyListItemList
```

See how changing the contents of item 4 of `deepCopy` does not modify `myList` (because `deepCopy` has duplicated all data), but changing item 5 of `myList` is reflected by both `myList` and `shallowCopy`, because that value was shared by both variables.

Copying and scalars

As discussed earlier in the article, care should be taken to remember that, that `set` performs a deep copy on scalar items.

Conclusions on Types of Copy

So when do we want a shallow copy, and when would we prefer a deep copy? In situations requiring speed, or recursion, a shallow copy is probably your best bet. A deep copy is useful when your script is going to be changing values of a list, but you need to leave the original values intact. Duplicating all this data has a price, and scripters must be aware what exactly `copy` is doing, why it takes so long, and the alternative.

Conclusion

This article has covered ***a lot*** of ground. From learning the three different classifications of AppleScript variable types (scalar types, container types, and examining the reference type in detail), examining the different ways of appending items to a list, to picking the correct copy type for the correct situation. Even without interacting with applications, AppleScript is a language with a lot of power, and (like real world human languages), different ways to express concepts. By understanding these concepts, the best possible solution can be chosen, and your scripts can run better and faster. A faster script means everybody wins: the scripter, the end users, and the overall system. Happy Scripting and best of luck on your further AppleScript projects!

References

AppleScript Language Guide, Apple Computer:
http://developer.apple.com/documentation/AppleScript/Conceptual/AppleScriptLangGuide/

AppleScript: The Definitive Guide, Matt Neuburg, ISBN: 0-596-00557-1:
http://www.oreilly.com/catalog/applescpttdg/

The author would like to take the time to thank (in alphabetical order): Jared Barden, Hamish Sanderson, and Matthew Strange for their review and technical insights during the creation of this article.

MT

About The Author

Ryan Wilcox is the founder of Wilcox Development Solutions (www.wilcoxd.com) specializing in carbonization, cross-platform application development and e-commerce solutions. While leaning mostly towards Python folk, he does sometimes get away to use AppleScript. He can be reached at rwilcox@wilcoxd.com

HostedStore™

1 HostedStore provides a **TURNKEY SOLUTION FOR CREATING A WEB STORE**, yet is extensible so developers can use the application as a base for creating customized solutions for their clients.

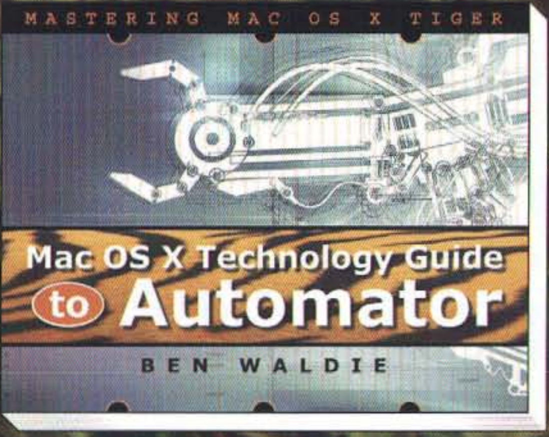
2 The extensibility of HostedStore provides a structure which allows developers to create modules that add new functionality or alter the built-in features.

3 We are seeking software resellers, hosting providers, developers, and designers that currently offer or plan to offer ecommerce solutions to their clients. Resellers receive a discount off of the published list price of our software and are free to bundle our software with their own products and/or services.

[www.hostedstore.com]

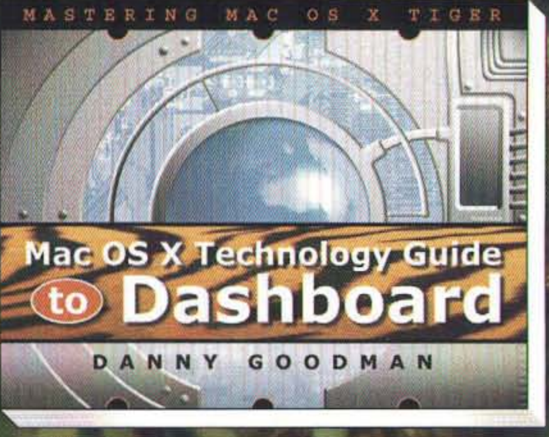
Lasso

Taming the Tiger


MASTERING MAC OS X TIGER

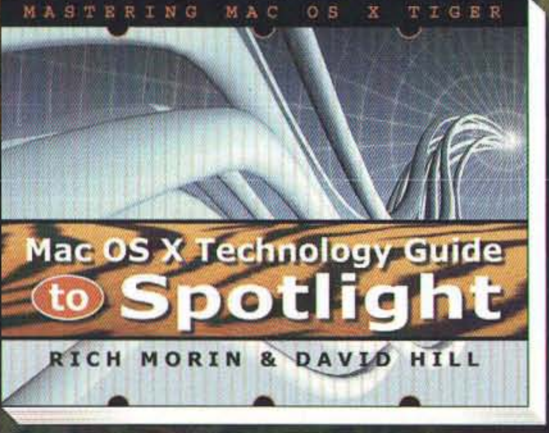
Mac OS X Technology Guide to Automator

BEN WALDIE


MASTERING MAC OS X TIGER

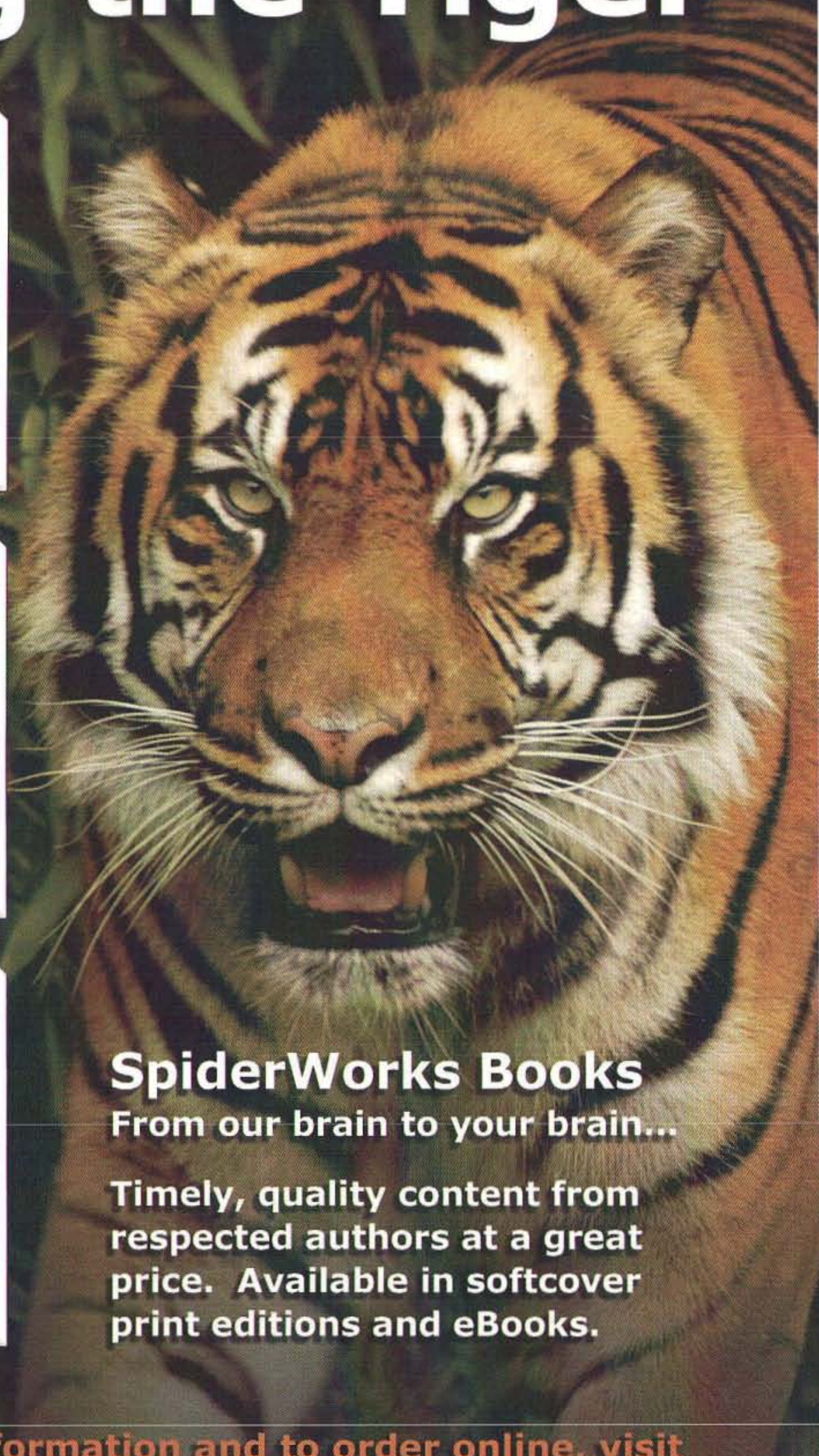
Mac OS X Technology Guide to Dashboard

DANNY GOODMAN


MASTERING MAC OS X TIGER

Mac OS X Technology Guide to Spotlight

RICH MORIN & DAVID HILL



SpiderWorks Books

From our brain to your brain...

Timely, quality content from respected authors at a great price. Available in softcover print editions and eBooks.

For more information and to order online, visit

www.spiderworks.com

STORING AND ACCESSING DATA WITH APPLESCRIPT

In last month's column, I provided an introduction to Database Events, a new technology that made its debut with Mac OS X Tiger. I discussed how Database Events can be used as a method of data storage and retrieval by allowing AppleScript to interact directly with SQLite databases.

In this month's column, I would like to discuss some other methods of storage and retrieval, such as accessing properties directly within scripts, or within property list files in the operating system.

Script Properties

The first mechanism for data storage that I would like to address is a ***script property***. When scripting an application, many times, a ***class*** within that application will possess properties, or attributes that are accessible through AppleScript. For example, in the Finder, the `disk` class possesses a number of properties, including a name, capacity, and format. When writing a script, you can define properties for your script itself. A script property is defined within a script using the following syntax:

```
property propertyName : «Property Value»
```

For example:

```
property theUserName : "bwaldie"
```

Script properties may be defined anywhere within the top level of a script (or a script object). Once a property has been defined, it becomes global in nature, and is accessible both throughout the top level of the script, as well as throughout any handlers in the script.

For information about script objects, please refer to the AppleScript Language Guide at:

http://developer.apple.com/documentation/AppleScript/.

When a script property is defined, its value is officially assigned when the script is compiled. This means that there is no need to define a value for the property, as you would a variable, within your script's executable code. Rather, you may immediately begin referring to the property.

Script properties are accessible in the same way that variables are accessible. You may get the value of a property and you may set the value of a property in the same way that you would do with a variable. Unlike local variables, however, properties are global in nature. This means that the property is accessible at any level of a script – at the top level, and also within any handlers. For example, after defining the property in the previous example code, the following code would be valid executable code at any level of the script, and would not generate an error indicating that `propertyName` is not defined.

```
display dialog propertyName
```

Properties also retain their most recent values between script executions, and will continue to do so until the script is recompiled. This makes properties an ideal mechanism for storing and retrieving data, so long as the script does not need to be frequently recompiled.

Working with a Property in a Script

Let's take a look at a property in action. The following example code demonstrates how a property can be used to store a persistent value between script executions.

```
property theRunCount : 0

set theRunCount to theRunCount + 1
display dialog "This script has been run " &
theRunCount & " time(s)."
```

If you run the previous code in Script Editor, the first time the script is run, a dialog will be displayed indicating that the script has been run one time. If you then run the script again, the dialog will indicate that the script has been run two times. If you run it again, the dialog will indicate that the script has been run three times. This will continue indefinitely until the script is recompiled. If you do recompile the script and run it again, the script will start over, indicating that it has been run one time.

To quickly recap what is occurring in this code, the first line of the script defines a property, named `theRunCount`. The second line changes the value of the property by incrementing its value by one. The third line displays a dialog message that includes the new property value. This new property value is then retained by the script until the next execution, when it is incremented again. When the script is recompiled, the original value of `0` is re-assigned to the property.

Properties are a great way to store information such as commonly requested file or folder paths, usernames, run counts, last execution dates, and more.

IMPORTANT: Please be aware that properties that are assigned in AppleScript Studio projects are NOT persistent between script executions.

Working with a Property in Another Script

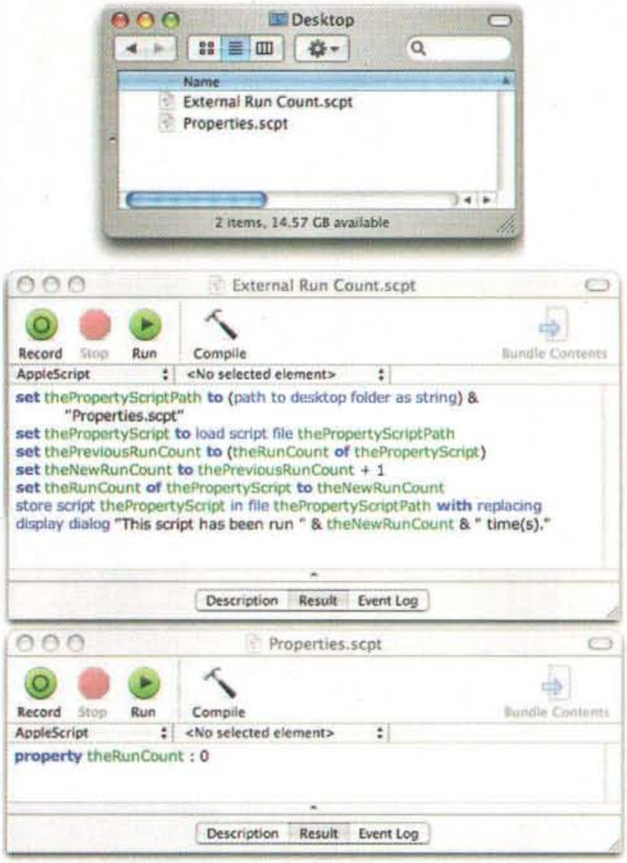
Now that we have discussed storing and accessing properties within a script, let's take the concept one step further – storing and accessing properties in another script. First, create a new document in Script Editor, and enter the following code:

```
property theRunCount : 0
```

Next, save the script to the desktop, and name it *Properties.scpt*. Now, create a new Script Editor document, and enter the following code:

```
set thePropertyScriptPath to (path to desktop
folder as string) & "Properties.scpt"
set thePropertyScript to load script file
thePropertyScriptPath
set thePreviousRunCount to (theRunCount of
thePropertyScript)
set theNewRunCount to thePreviousRunCount + 1
set theRunCount of thePropertyScript to
theNewRunCount
store script thePropertyScript in file
thePropertyScriptPath with replacing
display dialog "This script has been run " &
theNewRunCount & " time(s)."
```

Now save this second script to the desktop, and name it *External Run Count.scpt*. See figure 1.



External Run Count.scpt
Properties.scpt

```
set thePropertyScriptPath to (path to desktop folder as string) &
        "Properties.scpt"
set thePropertyScript to load script file thePropertyScriptPath
set thePreviousRunCount to (theRunCount of thePropertyScript)
set theNewRunCount to thePreviousRunCount + 1
set theRunCount of thePropertyScript to theNewRunCount
store script thePropertyScript in file thePropertyScriptPath with replacing
display dialog "This script has been run " & theNewRunCount & " time(s)."
```

```
property theRunCount : 0
```

Figure 1. Accessing a Property from an External Script

If you run the *External Run Count.scpt* script multiple times, you will encounter behavior similar to that of my earlier examples. The script will display a dialog indicating how many times the script has been triggered. However, this value is not being stored internally. Rather, it is being stored externally in another script.

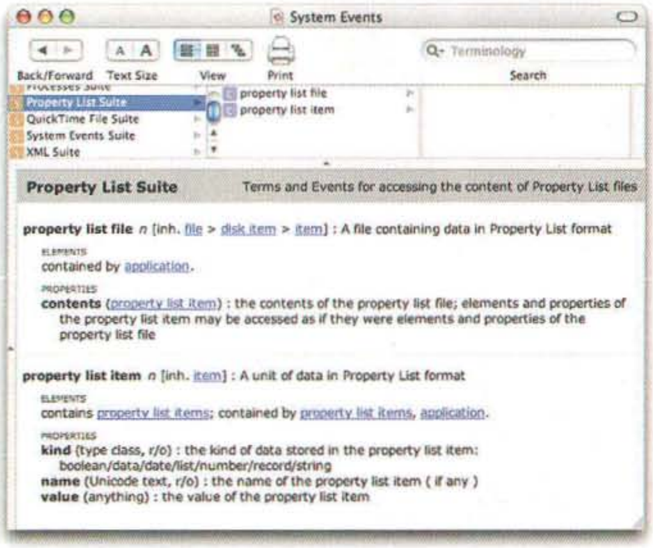
In the example above, the external script, *Properties.scpt*, is being loaded by the script *External Run Count.scpt*. The value of the property `theRunCount` is being retrieved, and changed within the loaded script. The modified loaded script is then being stored back into its original file, for the next execution.

There are many benefits to utilizing properties in this manner. Let's say, for example, that you have a script that you have created and saved as a run-only application. However, you want others to have the ability to change certain behavioral aspects of that script. You could create properties that control those behaviors, and store them in a separate, editable script. Then, your main script could load and access those properties as needed, during execution. I use this technique frequently when delivering scripts that may not have configurable user interfaces to clients. While the main code of my script may be locked, the client usually has the ability to change certain aspects of how the script behaves by modifying properties in an external script "settings" file.

Property List Files

The next mechanism that I would like to discuss for data storage and retrieval is property list files in Mac OS X, a.k.a. ***pList*** files. Property list files are XML-based files, which are utilized by many applications and processes in Mac OS X for storing and retrieving information.

In the past, I have mentioned an application named System Events, which is located in the ***System > Library > CoreServices*** folder in Mac OS X. System Events is a background application, which provides scriptable access to many aspects of Mac OS X. With the release of Mac OS X Tiger, Apple has introduced a new suite of terminology into the System Events application – a ***Property List Suite***. See figure 2.



Property List Suite Terms and Events for accessing the content of Property List files

property list file *n* [inh. file > disk item > item] : A file containing data in Property List format

ELEMENTS
contained by application.

PROPERTIES
contents (property list item) : the contents of the property list file; elements and properties of the property list item may be accessed as if they were elements and properties of the property list file

property list item *n* [inh. item] : A unit of data in Property List format

ELEMENTS
contains property list items; contained by property list items, application.

PROPERTIES
kind (type class, r/o) : the kind of data stored in the property list item: boolean/data/date/list/number/record/string
name (Unicode text, r/o) : the name of the property list item (if any)
value (anything) : the value of the property list item

Figure 2. System Events' Property List Suite

The *Property List Suite* in System Events provides AppleScript with a way to retrieve and modify values within property list files.

Creating a New Property List File

Before getting started with the actual scripting terminology, let me first mention creating property list files. Unfortunately, at present, System Events does not provide a way to create property list files via AppleScript. The intention at this point is to provide a way for scripters to interact with existing property list files.

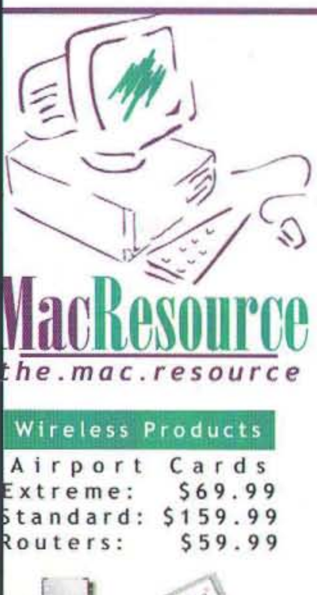
That said, inevitably, AppleScripters want the ability to create property list files. So, until that functionality is built into System Events (assuming it will be at some point in the future), here is some example code for creating a basic property list file with AppleScript. The code below will write the base XML of an empty property list file to a new text file on the desktop. Feel free to adjust this code as needed in order to meet your specific needs.

```
set theEmptyPListData to "<?xml version=\"1.0\"
encoding=\"UTF-8\"?>
<!DOCTYPE plist PUBLIC \"-//Apple Computer//DTD
PLIST 1.0//EN\"
\"http://www.apple.com/DTDs/PropertyList-1.0.dtd\">
<plist version=\"1.0\">
<dict/>
</plist>"

set theOutputFolder to path to desktop folder as
string
set thePListPath to theOutputFolder &
"myPListFile.plist"
set thePListFile to open for access thePListPath
with write permission
set eof of thePListFile to 0
write theEmptyPListData to thePListFile starting at
eof
close access thePListFile
```

Adding New Properties to a Property List File

If you need to generate property list files through scripting, then you probably also need the ability to add new properties to those files. Again, I'm sorry to say that, unfortunately, this is not the most straightforward process with System Events. As mentioned previously, System Events is intended to provide a way for you to be able to access and modify existing properties. At present, it is not really intended for creating property list files or adding new properties. However, there is a way to do it, and here it is.

In System Events, the *Property List Suite* consists of two classes, a `property list file` and a `property list item`. The `property list file` class has a `contents` property, which consists of a `property list item` class. This `property list item` class possesses `kind`, `name`, and `value` properties. Out of these, `value` is the only modifiable property. To add a new property list item within the contents of a property list file, you need to set the value of this property list item to an AppleScript record. This AppleScript record must consist of one or more field names and values, which correspond to



MacResource
the.mac.resource

Wireless Products

Airport Cards
Extreme: $69.99
Standard: $159.99
Routers: $59.99

Accessories

3RD PARTY Wireless
Mice $19 to $39
Apple Mice or Kbds
$19 to $59
Kensington 4-btn
Mice: $19

1-888-Mac-Resource

Parts for Every Mac System!

Logic Boards

G3 B&W Rev2: $99
G4 PCI : $129
G4 AGP Graphics: $219
G4 Gigabit: $349
G4 Digital Audio: $329
G4 Quicksilver: $499
G4 Mirrored Doors: $489
iMac Logics from $199
Beige G3 Series: $49

Power Supplies

Beige G3 Desktop: $49
Beige G3 Tower: $69
Blue & White Tower: $89
G4 Quicksilver: $179
G4 Digital Audio: $189
G4 Gigabit Ethernet: $199
G4 AGP Series 237w: $219
G4 AGP Series 208w: $179
G4 Mirrored Doors: $199

Logic board and power supplies require exchange.

Thousands of parts in stock for all Mac systems!

www.mac-resource.com

Mac Systems

G4/466Mhz: $249
G4/733Mhz: $699
G4/1000Mhz: $879
G4/1.25 OS9:$1099

New Systems Arriving Daily! Call for latest Stock.

eMac 700Mhz: $399
eMac 800Mhz: $449
eMac 1GHz: $629

iMac G4/700MHZ 15": $499
iMac G4/1.0Ghz 17": $749
iMac G4/1.25GHZ 20" $899

Xserves!!!

G4 1.0 ghz DP 512/60gb/CD NO SW $899
G5 2.0 ghz 512/80gb/CD $1999
G5 2.3 ghz DP 1024/80gb/Combo $2999
G5 2.3 ghz DP Nodes from $1999
Xserve Raid 1.2 TB 7 X 180gig $3399
Xserve Raid 2.5 TB 14 X 180gig $4499
Xserve Raid 2.8 TB 7 X 400gig $4999
Xserve Raid 5.6 TB 14 X 400gig $6999
Cache, Back-Up Batteries & Drive Upgrades available on all units
Free Ground Shipping when you mention MacTech Magazine

Refurb Displays

Apple 17-inch LCD: $349
Apple 15-inch Studio LCD: $149
Apple 22-inch Cinema: $999
Apple 17-inch Studio CRT (VGA OR ADC): $49

Products are open box, refurbished or demo call for more information

XML keys and values, respectively. For example:

```
set theOutputFolder to path to desktop folder as
string
set thePListPath to POSIX path of (theOutputFolder
& "myPListFile.plist")
tell application "System Events"
  tell property list file thePListPath
    tell contents
      set value to {|keyName|:"keyValue"}
    end tell
  end tell
end tell
```

After running the code above, a new property would be generated as XML within the referenced property list file, and would appear as follows:

```
<dict>
  <key>keyName</key>
  <string>keyValue</string>
</dict>
```

You can also use a similar technique to add to property list files that already contain existing properties. The following example code demonstrates how this can be done.

```
set theOutputFolder to path to desktop folder as
string
set thePListPath to POSIX path of (theOutputFolder
& "myPListFile.plist")
tell application "System Events"
  tell property list file thePListPath
    tell contents
      set previousValue to value
      set value to (previousValue &
{|keyName2|:"keyValue2"})
    end tell
  end tell
end tell
```

In the example code above, first, the existing value of the property list file's content is retrieved. Next, a new AppleScript record is appended to that value. The revised value is then reapplied to the property list file. The resulting XML of the property list file appears as follows:

```
<dict>
  <key>keyName</key>
  <string>keyValue</string>
  <key>keyName2</key>
  <string>keyValue2</string>
</dict>
```

In the previous two examples, I have demonstrated how to create property list items that contain string values. There are other types of values that can be created in property list files, including booleans, dates, lists, records, etc. Here, is another example of code that will append a new property to a property list file. This time, a record will be added.

```
set theOutputFolder to path to desktop folder as
string
set thePListPath to POSIX path of (theOutputFolder
& "myPListFile.plist")
tell application "System Events"
  tell property list file thePListPath
    tell contents
      set previousValue to value
      set value to (previousValue &
{|keyName3|:{subKeyName1:"subKeyValue1",
subKeyValue2:"subKeyValue2"}})
    end tell
  end tell
end tell
```

The above example code would change the contents of the property list file to appear as follows:

BMS

THE LAW OFFICE OF BRADLEY M. SNIDERMAN

Need help safeguarding your software?

If you're developing software, you need your valuable work protected with trademark and copyright registration, as well as Non Disclosure Agreements.

Then, when you are ready to sell it, you can protect yourself further with a licensing agreement.

I am an attorney practicing in Intellectual Property, Business Formations, Corporate, Commercial and Contract law.

Please give me a call or an e-mail. Reasonable fees.

23679 Calabasas Rd. #558 • Calabasas, CA 91302

PHONE 818-706-0631 **FAX** 818-706-0651 **EMAIL** brad@sniderman.com

PHONEPIPE™ VOIP Service

Residential Plans

The PhonePipe 500

$14.99

500 Minutes - US and Canada

*Additional Minutes 3.5 cents

The PhonePipe 900

$19.99

900 Minutes-US, Canada, Austria, Belgium, Chile, China, Denmark, France, Germany, Hong Kong, Ireland, Italy, Malaysia, Netherlands, New Zealand, Norway, Singapore, South Korea, Spain, Sweden, Switzerland, Taiwan, UK

*Additional Minutes 3.5 cents

The PhonePipe Unlimited

$24.99

Unlimited Minutes
US and Canada

The PhonePipe Unlimited International

$34.99

Unlimited Minutes-US, Canada, Austria, Belgium, Chile, China, Denmark, France, Germany, Hong Kong, Ireland, Italy, Malaysia, Netherlands, New Zealand, Norway, Singapore, South Korea, Spain, Sweden, Switzerland, Taiwan, UK

*All Plans Include Caller ID, Voicemail, and Three-Way Calling

Business Plans

Base Line

Starting at
$20.00 a Month

Includes:

- VM • CallerID
- 3-Way Calling
- Call Forwarding
- Call Waiting Plus

*Long Distance Starting at 2 Cents Per Minute

Enhanced Line

Starting at
$30.00 a Line

Includes:

- Basic Line Features
- Microsoft Outlook Integration
- Find Me-Follow Me
- Simultaneous Ring
- Personal Web Portal

*Long Distance Starting at 2 Cents Per Minute

Unlimited Line

$49.99 a Line

Includes:

- Enhanced Line Features
- Unlimited Long Distance to US and Canada

*Long Distance Starting at 2 Cents Per Minute

*All Business Lines Include PBX Features: Call Transfer, Music on Hold (Customizable), Call Hold, 4 or 5 Digit Dialing

Optional Features: Auto Attendant, Call Center, Reception Console

www.PhonePipe.com
1-877-300-3035 Ext 8200

```
<dict>
  <key>keyName</key>
  <string>keyValue</string>
  <key>keyName2</key>
  <string>keyValue2</string>
  <key>keyName3</key>
  <dict>
    <key>subkeyname1</key>
    <string>subKeyValue1</string>
    <key>subkeyvalue2</key>
    <string>subKeyValue2</string>
  </dict>
</dict>
```

Modifying Properties in a Property List File

Once properties exist within a property list file, the values of these properties may be modified using System Events. The following example code demonstrates how to modify the value of a property that was created in the previous example:

```
set theOutputFolder to path to desktop folder as
string
set thePListPath to POSIX path of (theOutputFolder
& "myPListFile.plist")
tell application "System Events"
  tell property list file thePListPath
    tell contents
      set value of property list item "keyName" to
"New Key Value"
    end tell
  end tell
end tell
```

Once the specified property has been modified, its new value will be reflected immediately in the property list file. For example:

```
<dict>
  <key>keyName</key>
  <string>New Key Value</string>
  <key>keyName2</key>
  <string>keyValue2</string>
  <key>keyName3</key>
  <dict>
    <key>subkeyname1</key>
    <string>subKeyValue1</string>
    <key>subkeyvalue2</key>
    <string>subKeyValue2</string>
  </dict>
</dict>
```

Likewise, property list items contained within other property list items may also be modified. In doing so, you must be sure to refer to these property list items within their proper containment hierarchy. The following example code demonstrates how to change the value of a property list item within another property list item.

```
set theOutputFolder to path to desktop folder as
string
set thePListPath to POSIX path of (theOutputFolder
& "myPListFile.plist")
tell application "System Events"
  tell property list file thePListPath
    tell contents
      set value of property list item "subKeyName1"
of property list item "keyName3" to "New Key Value"
    end tell
  end tell
end tell
```

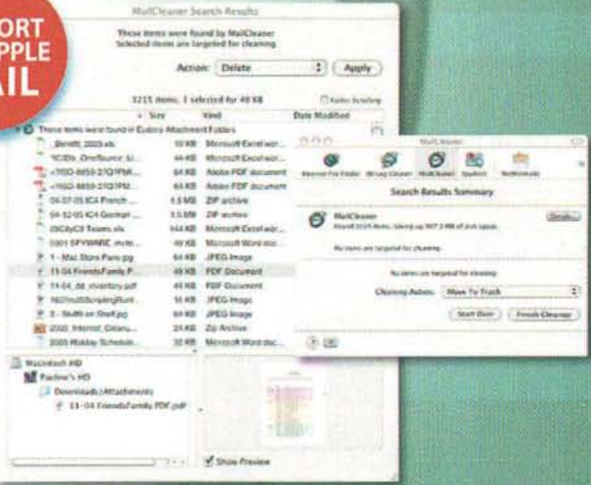
After running the previous code, the property list file's content would appear as follows:

```
<dict>
  <key>keyName</key>
  <string>New Key Value</string>
  <key>keyName2</key>
```

KNOW WHO'S WATCHING YOU.

PROTECT YOUR PRIVACY
MONITOR COMMUNICATIONS
SECURELY ERASE FILES
BLOCK ANNOYING WEB ADS

SUPPORT FOR APPLE MAIL



MailCleaner Search Results

These items were found by MailCleaner
Selected items are targeted for cleaning

Action: Delete — Apply

3215 items, 1 selected for 48 KB

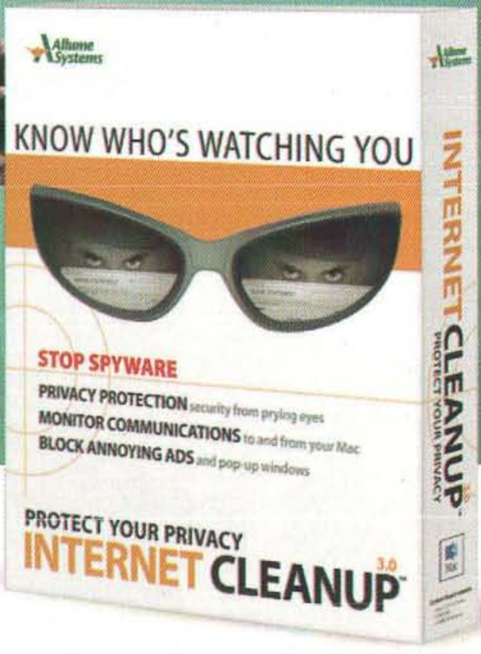
Size	Kind
10 KB	Microsoft Excel wor...
44 KB	Microsoft Excel wor...
64 KB	Adobe PDF document
64 KB	Adobe PDF document
1.5 MB	ZIP archive
1.5 MB	ZIP archive
144 KB	Microsoft Excel wor...
40 KB	Microsoft Word doc...
64 KB	JPEG Image
48 KB	PDF Document
48 KB	PDF Document
56 KB	JPEG Image
64 KB	JPEG Image
24 KB	Zip Archive
32 KB	Microsoft Word doc...

Macintosh HD
Downloads/Attachments
11-04 FriendsFamily PDF.pdf

Search Results Summary

MailCleaner

Cleaning Action: Move To Trash



Allume Systems

A Division of Smith Micro Software

Available from your favorite retailers and catalogs. Download a FREE Trial at Allume.com!

One hundred percent 10.4 "Tiger" compatible! Download a FREE trial version of Internet Cleanup at **http://www.allume.com/mac/cleanup/** today!

```
  <string>keyValue2</string>
  <key>keyName3</key>
  <dict>
    <key>subkeyname1</key>
    <string>New Key Value</string>
    <key>subkeyvalue2</key>
    <string>subKeyValue2</string>
  </dict>
</dict>
```

Retrieving Properties From a Property List File

As you might expect, retrieving a property value from a property list file is relatively straightforward. For example:

```
set theOutputFolder to path to desktop folder as
string
set thePListPath to POSIX path of (theOutputFolder
& "myPListFile.plist")
tell application "System Events"
  tell property list file thePListPath
    tell contents
      value of property list item "keyName"
    end tell
  end tell
end tell
--> "New Key Value"
```

One thing that I do want to stress is, when working with property list files, you do not need to access only property list files that you have created. You may actually use System Events to retrieve and/or modify the values of the property list files of other applications or processes.

For example, the code below demonstrates how to retrieve the value for one of Safari's preferences. In particular, this code will retrieve the value of the AlwaysShowTabBar preference, which indicates whether the tab bar should be displayed in Safari.

```
set thePListFolderPath to path to preferences
folder from user domain as string
set thePListPath to thePListFolderPath &
"com.apple.Safari.plist"

tell application "System Events"
  tell property list file thePListPath
    tell contents
      value of property list item
"AlwaysShowTabBar"
    end tell
  end tell
end tell
--> true
```

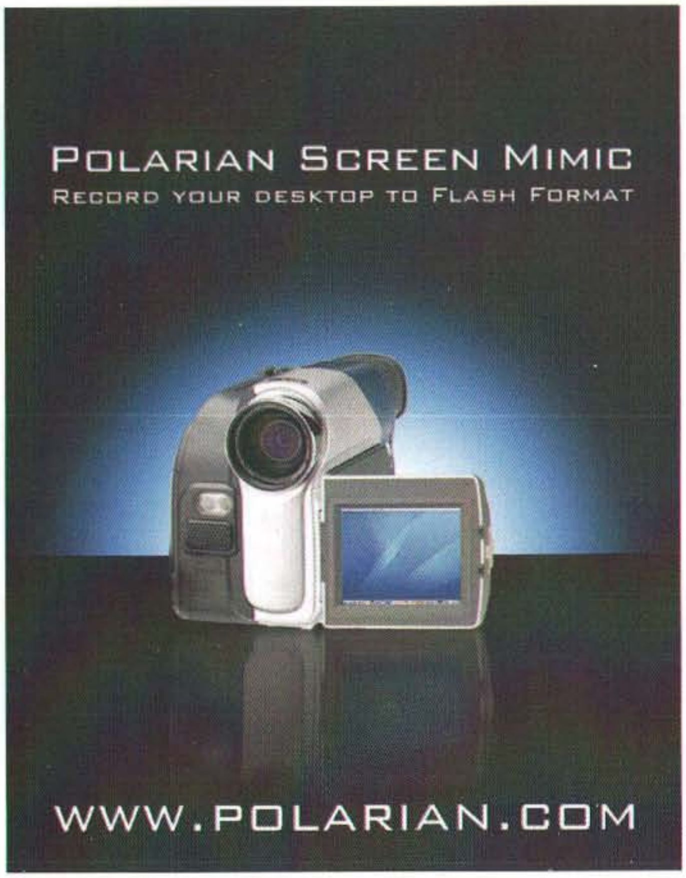
Storing Properties in an AppleScript Studio Project

Next, I'd like to touch briefly on AppleScript Studio. Earlier, I mentioned that AppleScript Studio does not have the ability to store persistent property values between script executions. However, in many cases, this may not be necessary anyway. The reason for this is because AppleScript Studio actually provides terminology for accessing properties in the application's property list file directly. This is done by accessing the user defaults class of the application class, which may contain default entry classes as elements. These default entry classes may be created and modified within the context of the user defaults class as needed during the execution of your script. Once the application quits, any default entries will be written to the property list file for the application, where they will be accessible the next time the application runs.

The following example code demonstrates how user defaults and default entries may be used to store information.

```
on launched theObject
  if (default entry "theRunCount" of user defaults
exists) = false then
    make new default entry at end of default
entries of user defaults with properties
{name:"theRunCount", contents:0}
  end if
  idle
end launched

on idle
  set theRunCount to contents of default entry
"theRunCount" of user defaults
  set theNewRunCount to theRunCount + 1
  display dialog "This script has been run " &
theNewRunCount & " time(s)."
  set contents of default entry "theRunCount" of
user defaults to theNewRunCount
  quit
end idle
```

In the on launched handler above, an if statement is used to determine whether a default entry named theRunCount already exists in the user defaults. If it does not, then it is created with an initial contents value of 0.



When the `on idle` handler triggers, the run count default entry is retrieved from the user defaults, incremented, and displayed in a dialog. The incremented run count is then applied back to the appropriate default entry in the user defaults. When the application quits, this information is stored in the property list file for the application. Therefore, if this application is triggered multiple times, the run count value will continue to increment with each new execution.

User defaults provide yet another way to store and access data during script execution, though only in AppleScript Studio projects.

Other Options

I would like to also take a moment to mention that there are other ways that you may choose to store and retrieve data, should you decide that the methods mentioned previously do not meet your needs in a particular situation. One way is to store data in a standard text file of some type. Typically, a delimited format of some type makes a good choice, although it may require that you write some fancy parsing code. Another choice could be to store your data in a database, such as FileMaker Pro. If you're using Mac OS X 10.4 or higher, then you could also consider exploring Database Events further, which was the topic of last month's column.

In Closing

Hopefully, I have been able to provide some insight into several possible methods of data storage. Obviously, you are certainly not bound to make use of only the techniques that I discussed. There are other mechanisms, which I encourage you to explore. However, the techniques that I have mentioned are the most commonly used among scripters, and generally do provide you with a variety of relatively quick and easy ways to store and retrieve data using AppleScript.

Until next time, keep scripting!

MT

About The Author

Ben Waldie is the author of the best selling books "AppleScripting the Finder" and the "Mac OS X Technology Guide to Automator", available from http://www.spiderworks.com. Ben is also president of Automated Workflows, LLC, a company specializing in AppleScript and workflow automation consulting. For years, Ben has developed professional AppleScript-based solutions for businesses including Adobe, Apple, NASA, PC World, and TV Guide. For more information about Ben, please visit http://www.automatedworkflows.com, or email Ben at applescriptguru@mac.com.

Long Distance

2.9¢ Per Minute!

Straight 6 second billing increments

Excellent rates on intrastate, intralata/toll calls and international calling with no term contract.

Toll Free (800/888/877/866) service, same low per minute rate for incoming calls.

10 cents per minute calling card.

Detailed billing directly from OPEX.

Quality electronic and telephone customer support.
No monthly billing fee if your bill is over $20.00 each month.

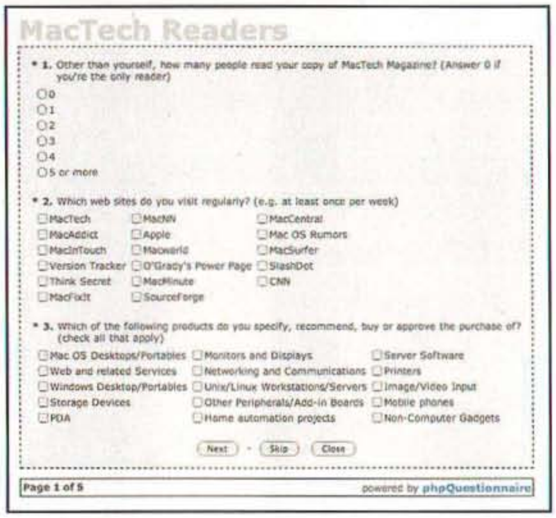
(NOTE: $2.00 billing fee is charged when your bill is under $20.00.)

www.utilities4less.com

Web Surveys: Quick and Easy

Have you ever needed to do a survey, quick ... but you didn't have time to go and custom code it for yourself. Recently, we had this need at MacTech ... you may remember a recent reader survey. We looked around and came across a neat product called phpQuestionnaire (or phpQ for short).

phpQ installed really easily on our Xserve running Mac OS X Server, and took advantage of our PHP 4 and mySQL setup. The installation instructions were a breeze, and the interface was fairly intuitive.



MacTech Readers

* 1. Other than yourself, how many people read your copy of MacTech Magazine? (Answer 0 if you're the only reader)

- ○ 0
- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5 or more

* 2. Which web sites do you visit regularly? (e.g. at least once per week)

☐ MacTech	☐ MacNN	☐ MacCentral
☐ MacAddict	☐ Apple	☐ Mac OS Rumors
☐ MacInTouch	☐ Macworld	☐ MacSurfer
☐ Version Tracker	☐ O'Grady's Power Page	☐ SlashDot
☐ Think Secret	☐ MacMinute	☐ CNN
☐ MacFixIt	☐ SourceForge	

* 3. Which of the following products do you specify, recommend, buy or approve the purchase of? (check all that apply)

☐ Mac OS Desktops/Portables	☐ Monitors and Displays	☐ Server Software
☐ Web and related Services	☐ Networking and Communications	☐ Printers
☐ Windows Desktop/Portables	☐ Unix/Linux Workstations/Servers	☐ Image/Video Input
☐ Storage Devices	☐ Other Peripherals/Add-in Boards	☐ Mobile phones
☐ PDA	☐ Home automation projects	☐ Non-Computer Gadgets

Page 1 of 5 powered by phpQuestionnaire

This product gives you a very detailed, web-based survey ability. The admin of it is, as you would expect, all web based, and the question editor is fast and easy to use. Only once in creating a survey did I pause to figure out how to do something.

You can run multiple surveys simultaneously and the phpQ takes care of tracking things separately. And, because it's template driven, you can change the look and feel if you so desire. (Although if you are lazy like I was being, you just take advantage of their professional templates ... after all, it was just a survey!!)

There are many different question types, and you can create surveys with all kinds of combinations. Everything I needed, it had built it ... radio buttons, check boxes, text boxes, text areas, etc...

You can use some simple restrictions or be more advanced ... to avoid bogus data, or duplicates. Unique IPs, cookies, and email verifications are all an option.

One of the nicest features of this product is it's ability to "preview" an entire survey before you publish it. It helped with layouts, groupings, page layouts and ordering. Again, it was fast and simple ... and by just opening the survey in a separate window, I could edit the survey in one window, while viewing it in another.

Once your users have entered the information, you can then view the results in a summarized, detailed or individual manner. And, as you would expect, you can export these results as well.

If you want the public to review the results, they can ... or you can keep them private. Your choice. You can also get notifications to admins when people are doing the survey.

Because this is all based on mySQL, you can go into mySQL and poke around if you'd like. But, with backup and restore features in phpQ, and the ability to export, there's really not much reason to.

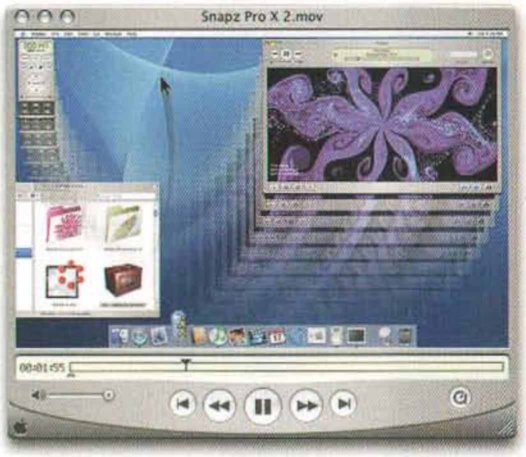
chumpsoft has done a bang up job here with phpQuestionnaire 2.2. The full version is $199, and there's a junior version for $59 along with some other flavors.

http://www.chumpsoft.com/

MT

Snapz Pro X 2.0

Move ahead of using static screenshots for conveying an idea effectively and take full motion video screen captures and create screen casts with Snapz Pro X 2.0. Snapz Pro X 2.0 is a great (and easy to use) screen capture utility for capturing moving or still pictures on Mac OS X.



Snapz Pro X 2.0 captures full motion video of anything on your Mac screen, along with digital audio and an optional microphone voice over at the rate of 30 frames per second in millions of colors. Snapz Pro X 2.0 is effectively a digital video camera, which allows recording anything on your Mac screen and saving it as a QuickTime movie or screen shot. A real-time preview of the screen cast is available. Screen casts can also be saved in a QuickTime compatible video compression format to optimize for data rate, frame rate, and color depth for creating smaller size videos.

Snapz Pro X 2.0 really simplifies such tasks as producing product demos, creating tutorials, making training videos, and archiving streaming video. If you are the type that wants to, you could even use Snapz Pro for showing their peers how good a game is or how to use an application. Snapz Pro X 2.0 features a completely redesigned interface for providing powerful features while maintaining elegant simplicity of Snapz Pro X. Snapz Pro X was completely rewritten for Mac platform.

If you are just looking for static screen captures, Snapz Pro X 2.0 packs in numerous new features. "Live Preview" is one of the most powerful and time saving features, which shows you exactly how your screen shot will look before you save it to disk, allowing on the fly change in border styles, scaling, cropping, and other settings. Another very useful feature is "Fatbits" tool that allows zooming in on the pixels on your Mac screen for automatically generating image thumbnails.

Screenshots can be scaled, cropped, color depth-changed, and dithered. Snapz Pro X 2.0 can also add borders, generate automatic thumbnails, and overlay watermarks/copyright notices. Snapz Pro X 2.0 uses Ambrosia's proprietary Clearscale technology to scale images up or down for both static images, and screen casts. Snapz Pro X 2.0 supports saving screen shots as .bmp, .pict, .gif, .jpg, .png, .tiff, .pdf, or Photoshop files, with precise control over image compression levels and color depth.

Snapz Pro X 2.0 costs $69 (screenshot only version priced at $29), with upgrades from Snapz Pro X 1.0 w/ movie capture priced at $20. Snapz Pro X 2.0 requires Mac OS X 10.2 or later

MT

KLEAR SCREEN

The Right Way to Clean Your Screen, iPods, and More...

If you've been looking for a way to clean your LCD, plasma, HDTV, flat screen, and CRT displays, then you may want to take a look at the products by Klear Screen. Klear Screen has designed, a set of products specifically to clean, protect, and polish all Apple displays and iPods, in addition to laptops, especially X-Brite or Crystal Clear Screens, new Generation LCD screens, plasma and HDTVs, digital cameras, and camcorders.

Klear Screen products contain a 3-step liquid polymer-based formula that lifts and dissolves surface contaminants and fingerprints, and floating debris off the screen surface, leaving an anti-static coating. The coating, significantly reduces surface friction and wear, resists fingerprinting, and provides a renewable protective screen barrier. Klear Screen products are non-toxic, and alcohol and ammonia free.

Klear Screen Deluxe Cleaning Kit is the recommended solution for your cleaning requirements. Klear Screen Deluxe Cleaning Kit costs $ 24.95. It includes:

1- 8 oz. Klear Screen Pump Spray
1- Large Micro-Chamois Polishing Cloth
6- Klear Kloths
2- Klear Screen Travel Singles
1- Travel Size Micro-Chamois

Klear Screen Deluxe Cleaning Kit

Klear Screen products that are a part of the Klear Screen Deluxe Cleaning Kit, are also available individually. The Klear Screen 8 oz. Pump Spray Bottle that lasts for approximately 1,200 cleanings, costs $12.95. Klear Screen Micro-Chamois Polishing Cloth, priced at $9.95, is a lint free, non-damaging, washable, and reusable cloth. Klear Screen Klear Kloths, also priced at $9.95, are made from lint-free, non-polyester, white-room, aerospace grade, non-woven polishing material that can be used and re-used for weeks at a time. Klear Screen Travel Singles are designed for the needs of individuals, who need a quick and easy-to-use cleaning solution whether at work, home, or in the classroom. Klear Screen Travel Singles cost $9.95. A pack of 3 Travel Size Micro-Chamois Polishing Cloths is priced at $9.95. Other products of the Klear Screen range include Klear Screen High performance Kit, Klear Screen Super Starter Kit, and Klear Screen Micro-Fiber Polishing Cloth.

MacTech's staff has been using Klear Screen products for years, and gives them a "two thumbs up". For more information, see http://www.klearscreen.com/.

MT

– by MacTech Staff

BLOB XML PHP C++
REALbasic Director COM .NET
SQL Object-Relational Client Server Unicode
Joining Worlds of Information

Valentina 2: Deploy True, Royalty Free Client-Server Solutions
www.paradigmasoft.com

PARADIGMA software

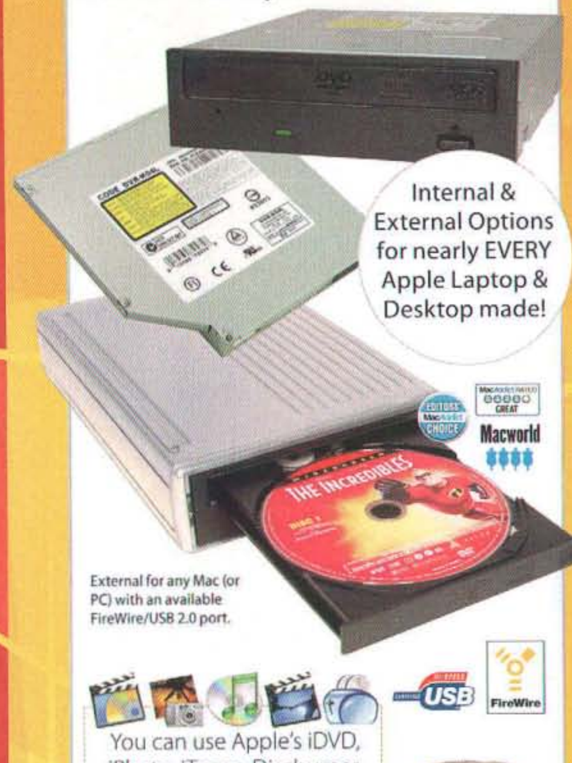
Other World Computing

Serving the Mac Universe since 1988

visit macsales.com 800.275.4576

SuperDrive Your Mac From Only $52.99!

Make Music, Movies & More Add a Fast SuperDrive to your Mac to Burn CDs, DVDs, even 8.5GB Dual Layer DVDs!



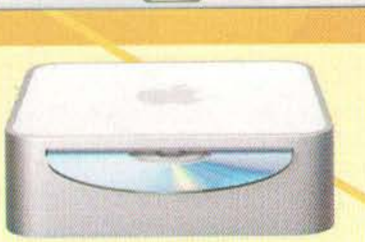
Internal & External Options for nearly EVERY Apple Laptop & Desktop made!

External for any Mac (or PC) with an available FireWire/USB 2.0 port.

You can use Apple's iDVD, iPhoto, iTunes, Discburner, etc; Roxio Toast; nearly any CD/DVD authorizing software available!

media, cases & label kits in stock too!

macsales.com/superdrive



Memory

More Memory for a Faster Mac!

From the min to the Max - OWC has the right Memory for your Apple.

For NEW 2006 MacBook Pro and iMac Intel models:
PC5300 DDR2 667MHz 200 Pin
512MB Modules only **$59.99**
1.0GB Modules only **$119.99**
2.0GB Set only **$239.00**

For PowerMac G5 and iMac G5 Models:
512MB DIMM from **$49.00**
1.0GB DIMM from **$92.99**
1.0GB Pair (512x2) from **$95.00**
2.0GB Pair (1GBx2) from **$185.00**
2.0GB DIMM from **$249.00**
4.0GB Pair (2GBx2) from **$495.00**

For PowerMac G4, iMac G4, eMac G4, Mac mini:
256MB from **$25.00**
512MB from **$49.00**
1GB from **$89.99**

For PowerBook 12/15/17; iBook G3/G4s:
512MB Modules from **$49.00**
1GB Modules from **$115.99**
2GB Sets from **$229.00**

Lifetime Advance Replacement Warranty.

Call 800-275-4576 or visit www.macsales.com/memory and use our online guide to get the right memory for your Mac!

Not sure what upgrade is best for your computer?

Then visit My OWC Upgrade Center and buy with confidence!

macsales.com/MyOWC

Hard Drives

Internal Hard Drives

Bulk up your computer by giving it higher capacity to perform for your needs.

For iMacs, eMacs & PowerMacs

3.5" Plug & Play 80GB to 500GB from $59

For PowerBooks, iBooks & Mac mini

2.5" 40GB & up to 160GB from $79.95

Not sure what your Mac takes? Buy with Confidence, call or use our online guides. **macsales.com/harddrives**

Hard Drive Controllers

Hard Disk Controller Cards

SIIG Inc.

Serial ATA 4-Channel PCI-M $79.95

SONNET

Tempo ATA & SATA Mac PCI Controllers
Tempo Trio FireWire/USB2/ATA-133
all in one PCI $149.00

FirmTek

4-Channel SATA Controllers
from $119.95

SATA Internal Mounting System

SONNET

Sonnet G5 Jive, add 3 SATA HDs to
PowerMac G5 $95.00

SATA External Enclosures

FirmTek

Dual-Bay Hot Swap External Serial ATA
Enclosure with SATA PCI Controller Card $257.95

SONNET

Sonnet Fusion 4-Bay SATA Kit $549.00

Hard Drive Enclosures

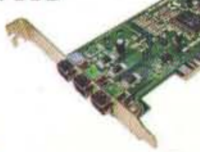
OWC

OWC Mercury Elite
SATA RAID Solutions

Perfect for Mass Storage
or RAID. Cables included.
Year OWC Warranty

Starting from
$79.95

ADD FIREWIRE & USB to your Mac!



FireWire from $9.95
FireWire+USB Combos from $29.99

PROSOFT

Drive Genius
Retail ~~$99.95~~ $65.00
Maintains, Manages, Optimizes, Repairs.
Keep your Hard Drives at their best. Recover data if disaster strikes!

Special OWC 35% Off! Now Only $65.00

Drive Genius
MacAddict RATED AWESOME

Highly Reliable, High-Performance, Plug & Play - FireWire & USB2 External Storage Solutions

Add up to 1 Terabyte - 1000GBs - for your Audio/Video, Backup, Music, Additional Storage Needs, etc with a top rated OWC or NewerTech Storage Solution. Exclusively using the latest Oxford Chipsets and the the latest Hitachi and Seagate drives, these solutions are built to last. All solutions are EMC/Dantz Retrospect Backup Certified and backed with up to a 2 Year OWC or NewerTech Warranty.

OWC Mercury Elite-AL Pro up to 500GB
FireWire 400+USB2 Solutions from $129.99
FireWire 800/400 Solutions from $149.99
FireWire 800/400/USB2 Solutions from $159.99
Add your own HD Enclosure Kits from $79.99

OWC Mercury Elite-AL 800 Pro RAID up to 1TB(1000GB)
FireWire 800/400 Solutions from $239.99
Add your own HD Two-Bay Enclosure Kits $119.99

MacAddict RATED GREAT · EDITORS' MacAddict CHOICE · Macworld · LAYERS

The ONLY Pocket-Sized Solution up to 160GB!

OWC Mercury On-the-Go Portable up to 160GB
FireWire 800/400/USB2 from $149.99
FireWire / FireWire 400/USB2 up to 120GB from $129.99
Add your own HD Enclosure Kits from $59.99

GEAR OF THE YEAR · Macworld · EDITORS' MacAddict CHOICE · MacGuide · Photoshop User

See the full line of OWC FireWire solutions, as well as solutions by LaCie, EZ Quest, SmartDisk, and Wiebetech online at macsales.com/FireWire

my mac 5 out of 5 · MacAddict RATED GREAT · MacAddict "DroolWorthy" · MacGuide

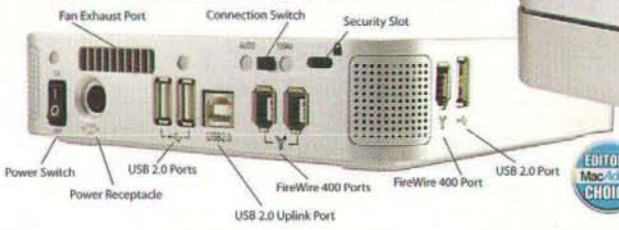
Neptune
FireWire 400 solutions from $99.95, Add your own HD Enclosure Kits just $47.99

EDITORS' MacAddict CHOICE · my mac 5 out of 5 · MacAddict RATED GREAT · Macworld · MacAddict "DroolWorthy"

OWC Mercury Elite Pro 'Classic' up to 400GB
FireWire 400+USB2 Solutions from $129.99
FireWire 800/400 Solutions from $159.99
FireWire 800/400/USB2 Solutions from $169.99
Add your own HD Enclosure Kits from $79.99

NewerTech miniStack™ *V2* FireWire+USB2 Solutions
with integrated FireWire and USB2 Hubs bring high performance storage and port convenience with solutions up to 500GB from $135.99
Add your own HD Enclosure Kits $79.95

miniStack™ *V2*



Macworld Magazine December 2005 'Top Product'.

EDITORS' MacAddict CHOICE · Macworld · MacAddict RATED GREAT

backup by EMC dantz Retrospect

Solutions include all cables, Dantz Retrospect Backup, Intech HD Speedtools and are preformatted with free bonus software loaded all ready to plug and play your PowerMac, PowerBook, iMac, iBook, or eMac too.

Mod your iPod

From High Capacity Batteries to cases, Other World Computing has you covered in keeping your iPod looking stylish as well as getting more playing time.

OWC's full line of iPod Batteries, Accessories, and more online at macsales.com/iPod

iPod Replacement Batteries & Enrichment Products

newertechnology

iPod Replacement Battery Kits
Easy to Install, Tools Included + Online Installation Videos. Get up to 78% more capacity & 20+ Hours Runtime!

iPod Batteries for nearly every Apple iPod *Starting From* **$14.99**

MacUser

Not comfortable opening your iPod? For **$39** + the cost of the battery, OWC installs it for you - iPod shipping Box and FedEx Overnight covered to and from! **macsales.com/iPodinstall**

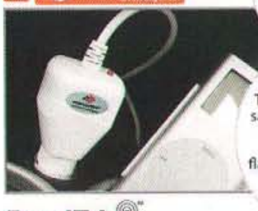
NuPower

iPod cases

RoadTrip & RoadTrip Plus
Listen to your iPod on-the-road!
Finally, an easy to use and GREAT sounding FM Transmitter for your iPod! Just plug, tune a single station and jam on without interruption. Easier to use and sounds better than products costing 2 times as much!

RoadTrip!+

RoadTrip!+ FM Transmitter + iPod Charger *Only* **$25.99**



Tom Keating of TMCnet.com had this to say about the RoadTrip!+ "The RoadTrip!+ FM Modulator worked flawlessly... and I was very impressed with the sound quality."

RoadTrip!

RoadTrip! FM Transmitter For ALL Apple iPods *Only* **$14.99**

Music on your Mac
macsales.com/music

M-AUDIO

Nova Large Capsule Cardioid Microphone **$99.00**

StudioPro 4 Desktop Audio Powered Monitors **$149.00**

Trigger Finger MIDI Controller Input Device **$199.00**

Ozone USB Audio 8 Midi Controller Knobs **$249.00**

contour

Shuttle A/V Controllers

Shuttle Pro v2 Jog/Shuttle (15 programmable buttons) **$84.99**

Shuttle Express Jog/Shuttle (5 programmable buttons) **$39.99**

ADS Tech Instant Music for Mac **$41.99**

elgato

EyeTV Video Encoder / TV Tuner/Digital Recorder **$149.99**
EyeTV 200 **$295.00**
EyeTV 500 High-Definition **$339.00**

FasterMac.net
Pay less. Get more. Surf faster!

Mac-Only Internet from only $5 per month!

High-Speed Nationwide Dial-up and DSL Services
Toll-Free Tech Support & More from Mac Experts

Visit FasterMac.net or call toll free 800-869-9152 to learn more or to sign up.

Software

Apple OS X 'Tiger' $99.00
full retail box version
OS X 10.2, 10.3 from **$17.99**

PostFacto

macsales.com/osx

NEW! iLife '06 $79.00

Apple iLife '06
Make the most out of your digital life. Share the magic of your everyday with iLife '06. Only **$79.00**

Mac Improvement

Laptop Batteries
newertechnology
NuPower
Batteries that Run Longer and Last Longer! Built in the USA and Built Right for up to 55% more runtime vs. your original Apple Stock Battery!

PowerBook G4 Ti from **$139.99**
PowerBook 12/15/17" from **$129.99**
iBook G3/G4 from **$119.99**

Call or Visit macsales.com/NewerTech

Laptop Screen Protectors
Protect your screen! There's an OWC Laptop Screen Protector (LSP) product for your Mac.
PowerBook G4 17" **$17.99**
PowerBook G4 15" **$17.95**
PowerBook G3 15" **$14.99**
iBook/PowerBook G4 12" **$13.95**

Stops marks!

The OWC LSPs are precision cut, glove soft leather protectors that prevent potentially permanent marks which can occur from the trackpad and keyboard while your laptop is closed.

Network Adapters
D-Link
D-Link 10/100 Ethernet PCI Card **$9.99**

PRAM Batteries
Is your Mac forgetting what time it is? OWC PRAM batteries starting at **$4.99**

Wireless Mouse
Logitech Cordless 'Click' Optical Mouse for USB SPECIAL $15.99

O'Reilly Books
The latest Mac titles from $9.95 Over 30 Mac Titles In Stock!

Mac mini upgrades

Upgrade to 1GB only **$93.95**
More Memory = Much Faster!

miniStack V2 From **$79.95**

Get a Bigger & Faster Hard Drive from **$79.99**
More Storage that's up to 43.8% Faster!

Burn DVDs & CDs for only **$119.99**
8X DVD Burner is twice as fast as Apple's current SuperDrive option + supports Dual Layer DVD Burning!

Exclusive OWC Online Video shows how to install these Mac mini upgrades or for **$99** including overnight pickup and return delivery, OWC will do the Mac mini upgrades for you!

Call or Visit macsales.com/macmini

The Latest Enhancements

Village Tronic VTBook
Add another CRT or Flat Panel Display to your Powerbook **$246.99**

iLugger iMac cases
for the iMac G5 or for Mac mini and/or up to 20" LCD Display 5 color combinations starting at **$99.95**

Rain Design i360°
A Turntable for your iMac G5 17" and 20" **$39.00**

Eye candies for your iMac. Six to choose from.

Explore with your G4

OWC is the Apple Upgrades Expert! We know how to make your Mac a Faster Mac. And it's really amazing too – with a new processor, your Mac can be like new again – even better than new as it's possible for it to be FASTER than even a brand new Mac too. With 30 Days to try and a full 100% refund if it's not for you – you've got nothing to lose except that spinning beach ball.

OWC Stocks the full line of G3 & G4 Processor-upgrades by these leading manufacturers:

G4 Single Upgrades from $195.00; G4/1.6GHz only $249.00
Or Crank it up as high as a Dual 1.8GHz for $595.00!

G4 Upgrades for PowerMac G3s, PowerMac G4s, Cube G4, PowerBook G3s –
Even Legacy PowerMac 7200-9600 Models!

Why Upgrade?

- **More Speed for Less Cost**
- **Use all your existing memory, peripherals, etc.**
- **Plug & Play and ZOOM**
- **Works with the Latest Software & OS X Tiger too!**
- **Makes your current Mac like new again!**

30 Day Money-Back on NewerTech and OWC brand upgrades!

Give us a call or check out our website. Our compatibility guide will show just what options are right to make your Power Mac, PowerBook, iMac, etc - a Faster Mac today!
800.275.4576 macsales.com/Faster

ATI MacEdition Performance Video Cards

When your Mac is Fast, Don't let a SLOW video card hold you back!

PCI Video Upgrade for Performance or Additional Displays:
ATI Radeon 9200 w/128MB for any Mac with an Available PCI Slot $127.95
Up to 2048x1536 resolution.
Compatible with up to 24" Displays!

High-Performance AGP Video Card Upgrades

All of the following support up to two displays
ATI Radeon x800 MacEdition with 256MB $449.00
TOP OF THE LINE G5 VIDEO Card supports up to Apple's 30" Display!
For PowerMac G5 Only.

For all PowerMac G4 Models:
ATI Radeon 9600 $79.00
Up to 256MB of High-Performance Video Memory
For 4X and 8X AGP G4, G5 models
Support for up to two Apple 30" Cinema Displays!
Quartz Extreme, Tiger Core Video Accelerated
For Power Mac G4 Digital Audio, QuickSilver, Mirrored-Drive Door, and all G5 Models.

NEW! ATI Technologies RADEON 9800 PRO MAC EDITION $259.00
Ultra High Performance Dual Head Video Card W/ DVI, VGA and S-Video Ports, 256MB. New, for PowerMac G4s.
Quartz Extreme, Tiger Core Video Accelerated

Other World Computing
Serving the Mac Universe since 1988
visit macsales.com 800.275.4576

Other World Computing
1004 Courtaulds Dr., Woodstock, IL 60098

DESKTOP SYSTEMS ENGINEER AND ANALYST

Interviewing Kevin Denges: Kevin is in charge of the image creation and deployment for Conde Nast. I spoke with him in New York City, at his office.

Schoun: Kevin, tell me about the setup.

Kevin: We have a Windows Active Directory infrastructure for authentication. The Macs are bound to the Active Directory server and the Xserves are used as image servers, primary and secondary K4 servers. K4 is our Adobe InDesign workflow. Each server is setup using 2 Xserve's, one primary and a second mirror for failover. We started all this with Mac OS X 10.3.3, so we've been at it for some time.

Schoun: And for file servers?

Kevin: We are using Extreme z-ip version 4. We started to deploy them in Sept/Oct of this year to our Windows servers, and by doing so, now have single sign on using Kerberos.

Schoun: What about the Macs? How are they deployed? NetBoot?

Kevin: No. The Xserves are just image servers for the most part. We use FireWire drives to deploy our images, but we also store those images on a server. So if a tech ever needs to get a newer image, they boot off of a FireWire drive and install the latest image from our servers.

Schoun: Sounds like NetBoot may solve the booting from FireWire issue.

Kevin: We are looking into that, but for right now, FireWire drives are the most efficient for us.

Schoun: So one image to rule them all?

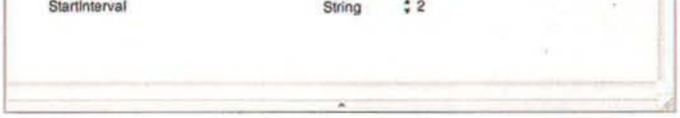
Kevin: Nope. We have an image that has all the software they need, and one main image. There are older images with 10.3 on them also. It depends on where the computer is placed and with whom as to what image they receive on their Mac.

Schoun: So the image build, it's stock Mac OS X?

Kevin: For the most part yes. We have leveraged the power of launchd to handle some initial binding and computer setup for us, but other than that, it's stock with Adobe's CS2 suite.

Schoun: Take us through the launchd file.

Kevin: Well basically it's very simple, it starts when the machine boots and runs a shell script that we have. Here's what it looks like:



Property List	Class	Value
▼Root	Dictionary	6 key/value pairs
Label	String	com.amg.adhook
LowPriorityIO	Boolean	Yes
Nice	Number	1
▼ProgramArguments	Array	1 ordered object
0	String	/Library/LoginWindow/ADhook.sh
RunAtLoad	Boolean	Yes
StartInterval	String	2

Figure 1.

Kevin: We also have another launchd item that handles the fixing of ByHost files.

Schoun: First tell me about the AD binding launchd item:

Kevin: As you can see, it runs at boot time, waits 2 seconds, then renices to get processor authority, to run ahead of other items. It then calls the shell script ADhook.sh located in a directory I chose. Pretty simple actually.

Schoun: Is the ByHost launchd item the same?

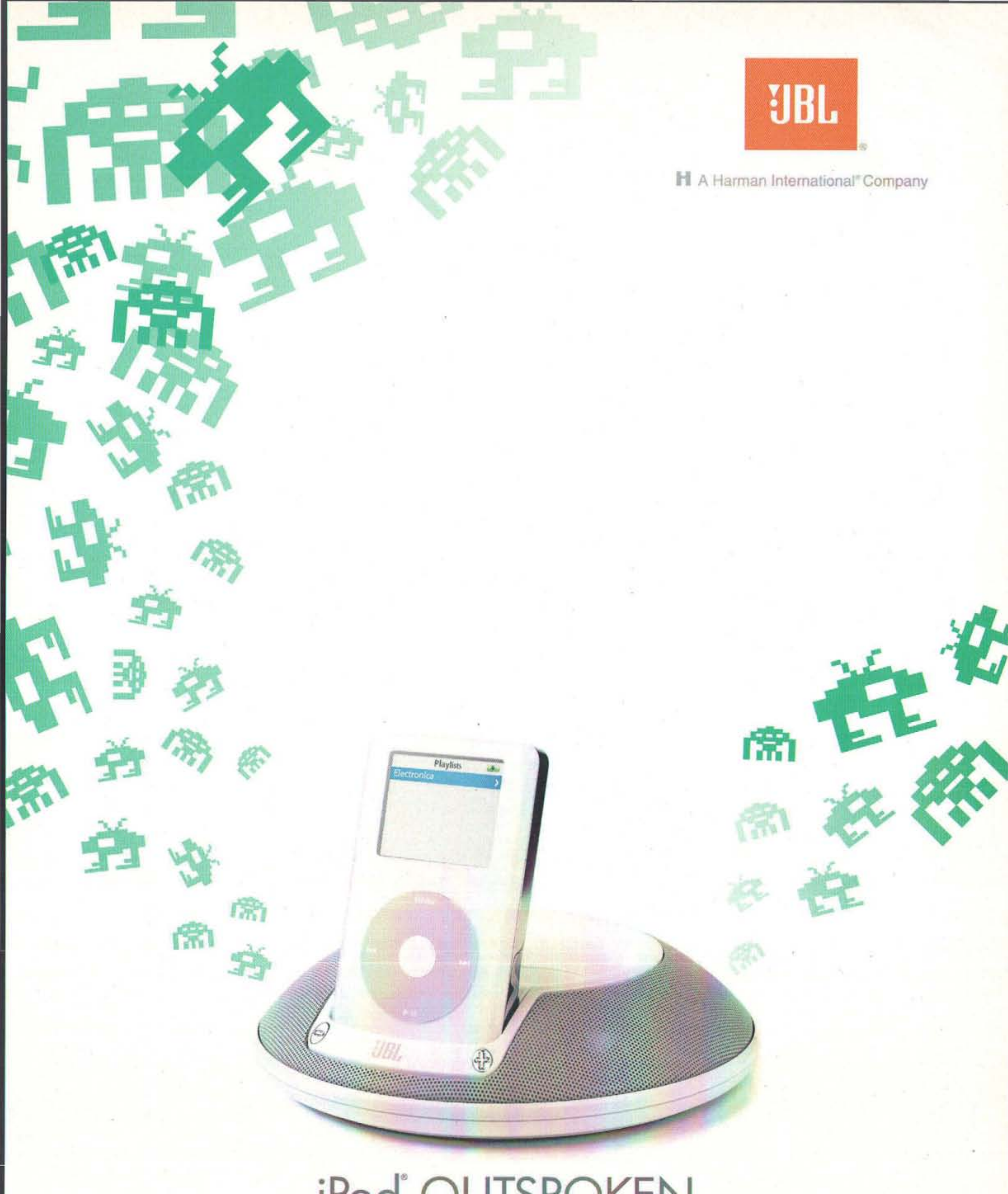
Kevin: For the most part yes, except it calls another script of course.

Schoun: So lets look at the AD script then. We know that the Active Directory plug-in has a command line counterpart, dsconfigad. I assume you use this?

Kevin: It's now a small part of the script, but yes, it is the

JBL

A Harman International® Company

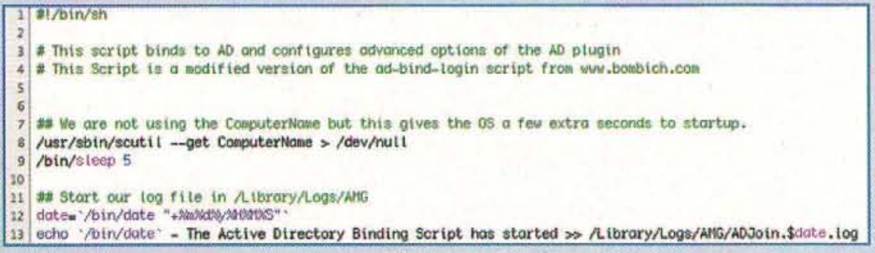


iPod® OUTSPOKEN.

JBL ON STAGE™: SHARE MUSIC OUT LOUD WITH THIS HIGH-PERFORMANCE SOUND SYSTEM FOR YOUR MP3 PLAYER.

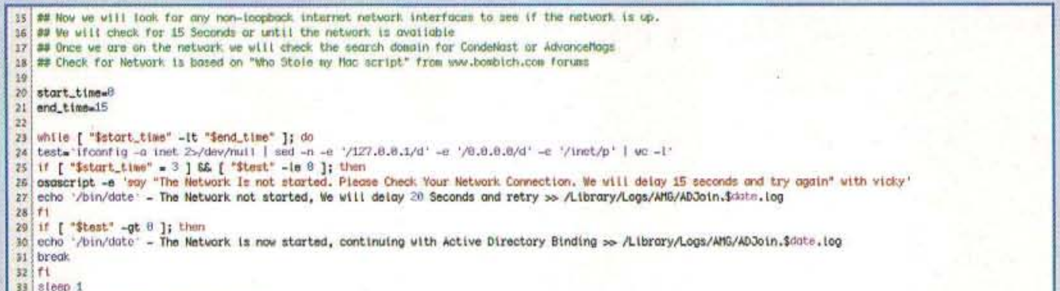
WWW.JBL.COM/JBLONSTAGE

Figures



```
#!/bin/sh

# This script binds to AD and configures advanced options of the AD plugin
# This Script is a modified version of the ad-bind-login script from www.bombich.com


## We are not using the ComputerName but this gives the OS a few extra seconds to startup.
/usr/sbin/scutil --get ComputerName > /dev/null
/bin/sleep 5

## Start our log file in /Library/Logs/AMG
date=`/bin/date "+%m%d%y%H%M%S"`
echo `/bin/date` - The Active Directory Binding Script has started >> /Library/Logs/AMG/ADJoin.$date.log
```

Figure 2.



```
## Now we will look for any non-loopback internet network interfaces to see if the network is up.
## We will check for 15 Seconds or until the network is available
## Once we are on the network we will check the search domain for CondeNast or AdvanceMags
## Check for Network is based on "Who Stole my Mac script" from www.bombich.com forums

start_time=0
end_time=15

while [ "$start_time" -lt "$end_time" ]; do
test=`ifconfig -a inet 2>/dev/null | sed -n -e '/127.0.0.1/d' -e '/0.0.0.0/d' -e '/inet/p' | wc -l`
if [ "$start_time" = 3 ] && [ "$test" -le 0 ]; then
osascript -e 'say "The Network Is not started. Please Check Your Network Connection. We will delay 15 seconds and try again" with vicky'
echo `/bin/date` - The Network not started, We will delay 20 Seconds and retry >> /Library/Logs/AMG/ADJoin.$date.log
fi
if [ "$test" -gt 0 ]; then
echo `/bin/date` - The Network is now started, continuing with Active Directory Binding >> /Library/Logs/AMG/ADJoin.$date.log
break
fi
sleep 1
 start_time=`expr "$start_time" + 1`
done
if [ "$test" -gt 0 ]; then
echo `/bin/date` - It took $start_time seconds to get a network connection>> /Library/Logs/AMG/ADJoin.$date.log
for net in $(ifconfig -l -u | awk '{ gsub ("lo0", ""); gsub ("fw1", ""); print}')
do
if [ "`(ipconfig getpacket ${net} | grep "yiaddr" | awk '{print $3}')`" == "" ]; then

   echo `/bin/date`  - "${net} - has no Valid IP Address" >> /Library/Logs/AMG/ADJoin.$date.log
   continue
   else
   echo `/bin/date`  - "${net} - has an IP Address of `(ipconfig getpacket ${net} | grep "yiaddr" | awk '{print $3}')`" >> /Library/Logs/AMG/ADJoin.$date.log
   DNS=`ipconfig getpacket ${net} | awk '/domain_name / {print $3}'`
   echo `/bin/date` - ${net} - has a search domain for ${DNS} >> /Library/Logs/AMG/ADJoin.$date.log
if [ -z ${DNS}]; then
osascript -e 'say "The Kevin Domain has not been found.  Binding to The Kevin Domain will probably fail." with vicky'
echo `date` " - Kevin domain name has not been found.  Bind to The Kevin domain will probably fail." >> /Library/Logs/AMG/ADJoin.$date.log
else
if [ ${DNS} = "kevinonedomain.com" ] || [ ${DNS} = "kevintwodomain.com" ]; then
osascript -e 'say "The Kevin Domain has been found.  Binding to The Kevin Domain is now starting." with vicky'
echo `date` " - The Kevin Domain has been found.  Binding to The Kevin Domain is now starting." >> /Library/Logs/AMG/ADJoin.$date.log
fi
fi
fi
done
fi;
```

Figure 3.

```
##### Fill in the AD plugin info here #####

## Check the AMG Logs folder to see if this script has been run before on this Mac.
logdir="/Library/Logs/AMG/"
list=$(ls -A1 $logdir 2>/dev/null | wc -l)

## We are going to get the MAC address of the computer to use in the AD name.
hwAddress=`/sbin/ifconfig en0 | awk '/ether/ { gsub(":", ""); print $2 }'`

##Get a UUID
uuid=`uuidgen | cut -d'-' -f5`

## If this Mac was already bound then use the UUID if not use the Mac Address
if [ "$list" -ge 2 ]; then
thename="$uuid"
else
thename="$hwAddress"
fi


## Set computer ID.

computerid="OSX-$thename"

## Other required AD plugin variables.
domain="kevinsadworkdomain.com"
id="macadbind"
password=`/usr/bin/osascript /Library/LoginWindow/ADP.scpt`

# Advanced options
alldomains="enable"         # 'enable' or 'disable' automatic multi-domain authentication
localhome="enable"          # 'enable' or 'disable' force home directory to local drive
protocol="smb"              # 'afp' or 'smb' change how home is mounted from server
mobile="enable"             # 'enable' or 'disable' mobile account support for offline logon
mobileconfirm="disable"     # 'enable' or 'disable' warn the user that a mobile acct will be created
useuncpath="enable"         # 'enable' or 'disable' use AD SMBHome attribute to determine the home dir
user_shell="/bin/bash"      # e.g., /bin/bash or "none"
preferred="-preferred secretname.kevinsadworkdomain.com"    # Use the specified server for all Directory lookups and authentication
                            # (e.g. "-nopreferred" or "-preferred ad.server.edu")
admingroups="KEVINSADWORKDOMAIN\domain admins, KEVINSADWORKDOMAIN\enterprise admins, KEVINSADWORKDOMAIN\client services level 1, KEVINSADWORKDOMAIN\client services level 2,
KEVINSADWORKDOMAIN\client services level 3, KEVINSADWORKDOMAIN\mac admin"            # These comma-separated AD groups may administer the machine (e.g. "" or "APPLE\mac admins")

##### End AD plugin variables. #####
```

Figure 4.

integral part. The script is divided into four major parts; network connectivity, AD binding, ByHost fixes, and file deletion.

Schoun: Can we walk through some of the major portions of the script?

Kevin: Sure, but keep in mind that some of this came from Mike Bombich's site, so credit goes to him for some of this.

Schoun: I'll let him know.

Kevin: So as we can see, the first part of the script creates a log file for us to write to, then we check for network connectivity.

See figure 2.

Schoun: I see Vicki in here.

Kevin: Yes. The problem we sometimes encountered was that a tech would download an image, and reboot the Mac only to find the Mac was not on the network. Or, in certain cases, the Macs would unbind and we would have to rebind them. Either way, the script now checks for not just network connectivity, but if the Mac is connected to our network. It announces its findings using the voice Vicki on the Mac. This way our techs know whether the machine is binding or not.

Schoun: Part of this script looks familiar.

See figure 3.

Kevin: [laughs] I borrowed-and gave credit to-Mike Bombich as this is based on the script called, "Who stole my Mac script". It was important to us, to make sure our techs knew what the status was, when they imaged a machine. This helps us track binding issues.

Schoun: Let's talk about the AD binding.

Kevin: Sure. Before I run

LOOK FAMILIAR?

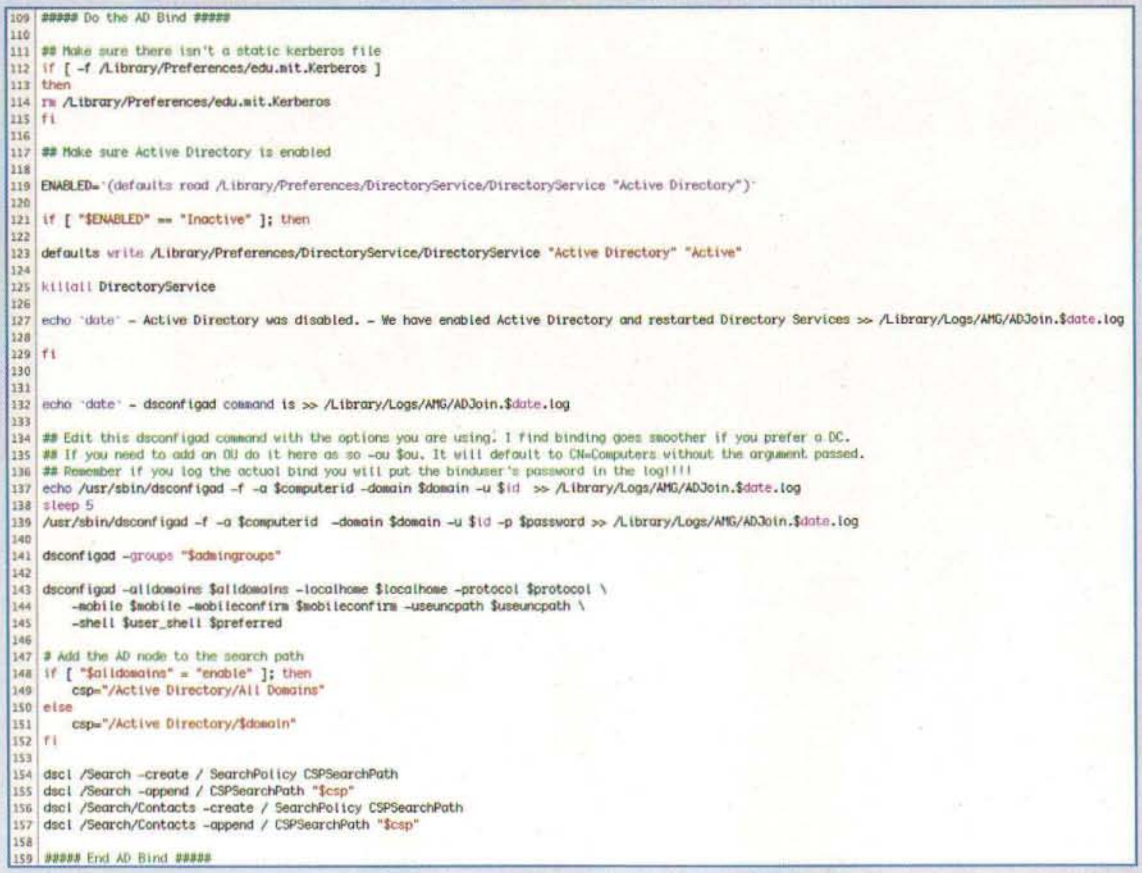
Get Rid of Your Fax Machine!

Receive your faxes directly to your email account as PDF file attachments

Service available in over 150 area codes

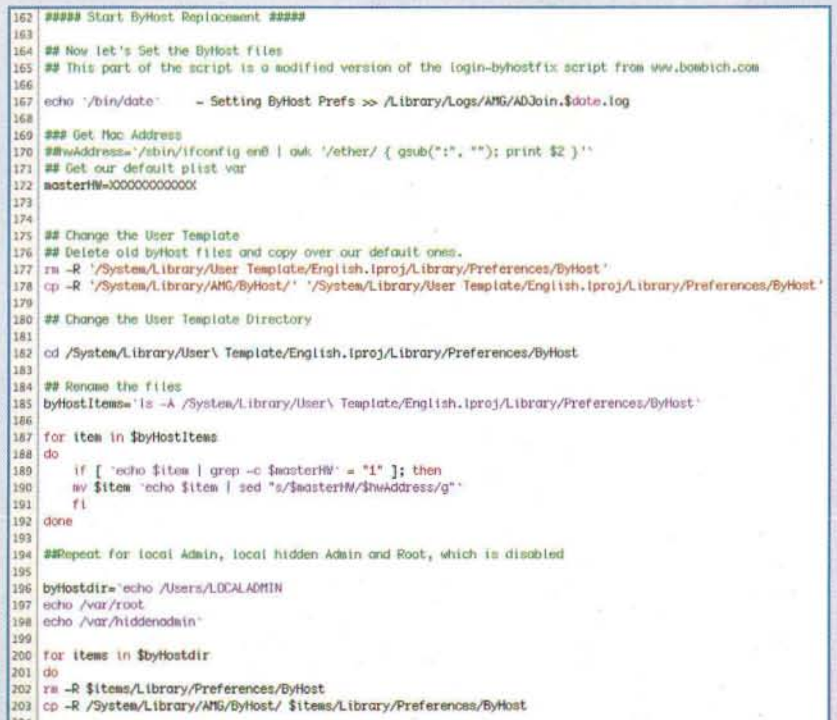
To check out special offers for MacTech readers, visit www.MaxEmail.com/MacTech

maxemail®



```
##### Do the AD Bind #####

## Make sure there isn't a static kerberos file
if [ -f /Library/Preferences/edu.mit.Kerberos ]
then
rm /Library/Preferences/edu.mit.Kerberos
fi

## Make sure Active Directory is enabled

ENABLED=`(defaults read /Library/Preferences/DirectoryService/DirectoryService "Active Directory")`

if [ "$ENABLED" == "Inactive" ]; then

defaults write /Library/Preferences/DirectoryService/DirectoryService "Active Directory" "Active"

killall DirectoryService

echo `date` - Active Directory was disabled. - We have enabled Active Directory and restarted Directory Services >> /Library/Logs/AMG/ADJoin.$date.log

fi


echo `date` - dsconfigad command is >> /Library/Logs/AMG/ADJoin.$date.log

## Edit this dsconfigad command with the options you are using. I find binding goes smoother if you prefer a DC.
## If you need to add an OU do it here as so -ou $ou. It will default to CN=Computers without the argument passed.
## Remember if you log the actual bind you will put the binduser's password in the log!!!!
echo /usr/sbin/dsconfigad -f -a $computerid -domain $domain -u $id  >> /Library/Logs/AMG/ADJoin.$date.log
sleep 5
/usr/sbin/dsconfigad -f -a $computerid  -domain $domain -u $id -p $password >> /Library/Logs/AMG/ADJoin.$date.log

dsconfigad -groups "$admingroups"

dsconfigad -alldomains $alldomains -localhome $localhome -protocol $protocol \
     -mobile $mobile -mobileconfirm $mobileconfirm -useuncpath $useuncpath \
     -shell $user_shell $preferred

# Add the AD node to the search path
if [ "$alldomains" = "enable" ]; then
     csp="/Active Directory/All Domains"
else
     csp="/Active Directory/$domain"
fi

dscl /Search -create / SearchPolicy CSPSearchPath
dscl /Search -append / CSPSearchPath "$csp"
dscl /Search/Contacts -create / SearchPolicy CSPSearchPath
dscl /Search/Contacts -append / CSPSearchPath "$csp"

##### End AD Bind #####
```

Figure 5.



```
##### Start ByHost Replacement #####

## Now let's Set the ByHost files
## This part of the script is a modified version of the login-byhostfix script from www.bombich.com

echo `/bin/date`      - Setting ByHost Prefs >> /Library/Logs/AMG/ADJoin.$date.log

### Get Mac Address
##hwAddress=`/sbin/ifconfig en0 | awk '/ether/ { gsub(":", ""); print $2 }'`
## Get our default plist var
masterHW=XXXXXXXXXXXX


## Change the User Template
## Delete old byHost files and copy over our default ones.
rm -R '/System/Library/User Template/English.lproj/Library/Preferences/ByHost'
cp -R '/System/Library/AMG/ByHost/' '/System/Library/User Template/English.lproj/Library/Preferences/ByHost'

## Change the User Template Directory

cd /System/Library/User\ Template/English.lproj/Library/Preferences/ByHost

## Rename the files
byHostItems=`ls -A /System/Library/User\ Template/English.lproj/Library/Preferences/ByHost`

for item in $byHostItems
do
     if [ `echo $item | grep -c $masterHW` = "1" ]; then
     mv $item `echo $item | sed "s/$masterHW/$hwAddress/g"`
     fi
done

##Repeat for local Admin, local hidden Admin and Root, which is disabled

byHostdir=`echo /Users/LOCALADMIN
echo /var/root
echo /var/hiddenadmin`

for items in $byHostdir
do
rm -R $items/Library/Preferences/ByHost
cp -R /System/Library/AMG/ByHost/ $items/Library/Preferences/ByHost

cd $items/Library/Preferences/ByHost
byHostItems=`ls -A $items/Library/Preferences/ByHost`
for item in $byHostItems
do
     if [ `echo $item | grep -c $masterHW` = "1" ]; then
     mv $item `echo $item | sed "s/$masterHW/$hwAddress/g"`
     fi
done
done
```

Figure 6.

the actual bind, I set all the parameters. This part of the script sets the parameters.

See figure 4.

Schoun: I see the uuidgen command in there! I should explain to our readers that uuidgen is a command that creates a unique ID, unique enough that the man page states, "The uuidgen command generates a Universally Unique Identifier (UUID), a 128-bit value guaranteed to be unique. A UUID is made unique over both space and time by combining a value unique to the computer on which it was generated—usually the Ethernet hardware address—and a value representing the number of 100-nanosecond intervals, since October 15, 1582 at 00:00:00." You gotta love Apple code writers sometimes. "Unique over space AND time" Ha.

Kevin: Well, the issue is that if a Mac binds to an AD domain, the AD domain cannot use that computer name to bind more than twice. So, if a computer's name is already in the AD domain, and our techs re-image it, the bind fails. Using a portion of the uuidgen output insures us a unique ID with which to bind the Mac.

Schoun: I also see that you did not put the password into this script for the actual bind. That you are calling the output of an AppleScript file. Why?

Kevin: Security. This way the script calls the AppleScript file and it returns the password. Just a little more secure this way.

Schoun: So with all the variables set, now the bind?

Kevin: Yes. Again we used the dsconfigad

Be the one with all the answers.

See the new MacTech Magazine.

TECH
The Journal of Macintosh Technology and Development
MAC OS X
ECURITY
ADHOC 2005: End of An Era?
Intro to Scripting Mail
MACTECH
The Journal of Macintosh Technology
IPOD-LINUX: HACKER'S DELIGHT OR PRIME TIME PLAYER?
BY EMMANUEL STEIN
BUILD A MAC MINI BASED ROBOT...THAT CAN SEE!
LIVING IN A VIRTUAL WORLD - TERMINAL APP FOR SYSADMINS
KNOCK KNOCK KNOCKING ON LDAP'S DOOR
ENTERPRISE BACKUP & RECOVERY - STREAMLINE LARGE-SCALE IMPLEMENTATIONS
Filter Your Email On OS X Server - With Open Sou
Bonjour From The Source Hound - Automatic Service Discovery in Tiger
X11 & OS X - How To Configure and Run Your Apps
Mac In The Shell - Back To bash Basics All Over Again!
QuickTime Toolkit - Working with Properties
AppleScript Essentials - Intro to Scripting iCal
Fetch, Clarus, Fetch - Core Data Part III
VODCASTING

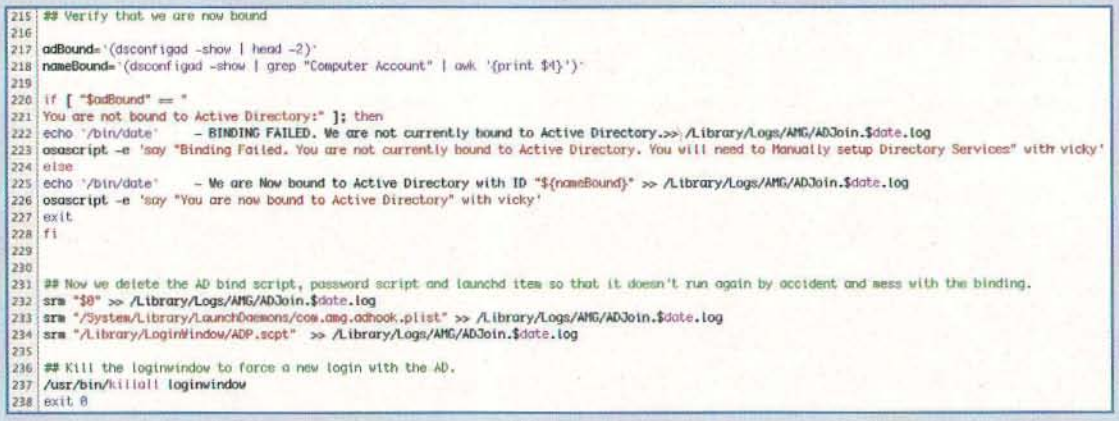
Make your Mac fly!

Get under the hood.

MacTech is already read every month by tens of thousands of readers. MacTech readers represent the very heart and soul of the the Mac community. Join the crowd and sign up for MacTech today!

For a one year subscription visit:
www.mactech.com/riskfree

Phone: 877-MACTECH / 805-494-9797



```
## Verify that we are now bound

adBound=`(dsconfigad -show | head -2)`
nameBound=`(dsconfigad -show | grep "Computer Account" | awk '{print $4}')`

if [ "$adBound" == "
You are not bound to Active Directory:" ]; then
echo `/bin/date`     - BINDING FAILED. We are not currently bound to Active Directory.>> /Library/Logs/AMG/ADJoin.$date.log
osascript -e 'say "Binding Failed. You are not currently bound to Active Directory. You will need to Manually setup Directory Services" with vicky'
else
echo `/bin/date`     - We are Now bound to Active Directory with ID "${nameBound}" >> /Library/Logs/AMG/ADJoin.$date.log
osascript -e 'say "You are now bound to Active Directory" with vicky'
exit
fi


## Now we delete the AD bind script, password script and launchd item so that it doesn't run again by accident and mess with the binding.
srm "$0" >> /Library/Logs/AMG/ADJoin.$date.log
srm "/System/Library/LaunchDaemons/com.amg.adhook.plist" >> /Library/Logs/AMG/ADJoin.$date.log
srm "/Library/LoginWindow/ADP.scpt"  >> /Library/Logs/AMG/ADJoin.$date.log

## Kill the loginwindow to force a new login with the AD.
/usr/bin/killall loginwindow
exit 0
```

Figure 7.

Vital Stats

Years in IT industry : 6

Information: Kevin is in charge of the image creation and deployment for Conde Nast.

Computers: About 2700 Macs, 42 Xserves, 4 Xserve RAIDs, Windows Active Directory Servers, wireless networks.

Programming Languages:Shell Scripting, AppleScript

command line tool for that. We of course write to the log file, check for an older edu.mit.Kerberos file and get rid of it, if one exists, and set the search policies.

See figure 5.

Schoun: I see a sleep command in there.
Kevin: It seemed to go smoother when we did this.
Schoun: Then the ByHost issues? Why the problem?
Kevin: As you know, ByHost files are MAC address specific, so we have them set the way we want, then we put all Xs in the place of the address. We then copy these and replace the Xs with the local Mac's MAC address. We do this for the template for any user's that log in via AD, our local administrator account, and a hidden account we have.

See figure 6.

Schoun: And the last part of the script?
Kevin: Checking the bind, having Vicki notify the tech, and then deleting the launchd item, the AppleScript file with the password, and then deleting itself, killing the loginwindow process and having it come back. This way, the AD user can log in and get a newly created home folder, with all the customized tweaks we've added in the user template, such tweaks to the .GlobalPreferences file, the Dock, the menu bar, the screen saver, the background, and so on.

See figure 7.

Schoun: This is in incredibly powerful script. Again I see some of Mike's work in here, but you have done a nice job of pulling it all together.
Kevin: Thanks. Our next step is to take this and make a launchd item that watches the bind and if it fails, it deletes the /Library/Preferences/edu.mit.Kerberos file, the DirectoryServices directory inside of the same location, and pulls them from a hidden location on the local disk and rebinds the machine automatically.
Schoun: I would also suggest, you use the mail command to email an administrator account, when this occurs. In this way, your techs can be a bit more proactive than reactive. They will know when a machine unbinds.
Kevin: Yea. We want to do more with launchd and scripts.
Schoun: Kevin, this is all we have time for, but I know you have more to say about your setup. I think this script is an excellent place to start. It gives the reader a clear- cut way of implementing this type of bind. I'd like to talk to you more about how this interacts with the local NetInfo database and what other tweaks you've done to the system.
Kevin: Be happy to. We are very confident in our process, and have moved to cut down time to under 5 minutes, if possible.
Schoun: I think our readers will be excited to learn about what else you have done.
Ed Note: Kevin has much more to say about his infrastructure but this time I wanted to focus on the script itself.

MT

About The Author

Schoun P. Regan is CEO of ITInstruction.com, which specializes in Mac OS X training and consulting. He speaks regularly to CEOs and CFOs on how to control IT department spending, the myths surrounding cross-platform integration, and the lunacy of expected lost revenue stemming from a culture bred to tolerate IT staff and operating system inadequacies as "normal". He seeks to change self-fulfilling IT departments that breed complacency for their jobs and contempt for the end user, neither of which are conducive to business.

Sidewinder™

www.MARWARE.com

Includes:
iPod case with
Slide-out cord wrap
Removable belt clip

Sidewinder™
for iPod nano & video

Smart armor

You didn't compromise on your music player—
don't compromise on your case.

kickstand for video viewing

What's the *difference*?

Brand new. It's perfect.
It's beautiful...

and it's *vulnerable*!

It's naked, exposed to
the elements.

It's...

INHUMANE

and it *will*
get scratched.

Brand new. It's perfect.
It's beautiful...

and it's **protected**!

Invisible, durable, &
rugged protection.

SAVE the iPod

www.the***invisible*****SHIELD**.com

Be the one with
all the answers.

MacTech is already read every month by tens of thousands of readers. MacTech readers represent the very heart and soul of the the Mac community. Join the crowd and sign up for MacTech today!

store.mactech.com/riskfree

Advertiser/Product Index

4D, Inc. 35
Absoft Corporation 33
Aladdin Knowledge Systems, Inc. IFC
Allume Systems, Inc. 7
Allume Systems, Inc. 29
Allume Systems, Inc. 63
Allume Systems, Inc. BC
Andescotia LLC 38
Brad Sniderman 60
Brian Loomis 54
CRYPTOCard Corporation 9
Daystar Technology IBC
FairCom Corporation 1
GLUON/CaptureWorks 25
Harman Multimedia 73
IGC/MaxEMail.com 75
Intego 23
I.R.I.S. Group USA 2
Kerio Technologies, Inc. 51
LTA Projects 10
MacResource Computers 59
MacScripter 22
MacSpeech, Inc. 40
MacTech Cables 20
MacTech Domains 57
MacTech Magazine 77
MacTech Printer Supplies 62
MARWARE, Inc. 79
Maxum Development 41
Meta Communications 47
MYOB US, Inc. 19
Netopia, Inc. 45
OmniPilot Software, Inc. 31
Other World Computing 68-71
Ovolab 15
Paradigma Software 67
Peachpit Press 39
Plantronics 53
Polarian Technology 64
Protective Solutions, Inc. 79
QuickerTek 52
RadTech 34
Seapine Software, Inc. 21
SpiderWorks 55
SubRosaSoft.com, Ltd. 11
Tellurium Communications 61
Utilities4Less.com 65

Absoft Compilers • Absoft Corporation 33
Accelerators/Upgrades • Daystar Technology IBC
AccountEdge • MYOB US, Inc. 19
Andescotia LLC • Andescotia 38
AquaZone • Allume Systems, Inc. 7
Bags • LTA Projects 10
c-tree Plus • FairCom Corporation 1
Cables • MacTech Cables 20
CaptureWorks • GLUON/CaptureWorks 25
CopyCatX/FileSalvage • SubRosaSoft.com, Ltd. 11
CRYPTOCard Corporation 9
Digital Rights Management • Aladdin Knowledge Systems, Inc. IFC
Domain Registration • MacTech Domains 57
Digital Storage Manager • Meta Communications 47
Harman Multimedia • Harman Multimedia 73
Hosted Store • Brian Loomis 54
iListen • MacSpeech, Inc. 40
InstallerMaker, StuffIt • Allume Systems 29
Internet Cleanup • Allume Systems, Inc. 31
Kerio Mail Server • Kerio Technologies, Inc. 51
Lasso • OmniPilot Software, Inc. 31
Law Offices • Brad Sniderman 60
Long Distance Phone Service • Utilities4Less.com 65
MacResource Computers • MacResource Computers 59
MarketBlast • 4D, Inc. 35
MacScripter.net • MacScripter 22
MacTech Magazine • MacTech Magazine 77
MaxEMail.com • IGC/MaxEMail.com 75
Other World Computing • Other World Computing 68-71
Peachpit Press • Peachpit Press 39
Phlink • Ovolab 15
PhonePipe • Tellurium Communications 61
Plantronics • Plantronics 53
PowerBook Accessories • QuickerTek 52
Print Supplies • MacTech Printer Supplies 62
RadTech 34
Readiris Pro • I.R.I.S. Group USA 2
Rumpus • Maxum Development 41
Screen Mimic • Polarian Technology 64
Screen Protection • Protective Solutions, Inc. 79
SpiderWorks ebooks • SpiderWorks 55
StuffIt Deluxe • Allume Systems, Inc. BC
Test Track Pro • Seapine Software, Inc. 21
Timbuktu • Netopia, Inc. 45
Trail Vue • MARWARE, Inc. 79
Valentina 2 • Paradigma Software 67
VirusBarrier • Intego 23

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

How to Stop Racing the Clock.

☐Work Longer? ☐Work Harder?

☑ Energize Your Mac!

We know your day keeps getting longer and longer. With every release of software, your Mac is bogged down even more. With every click, there's a pause. You find yourself working longer, working harder. *A faster Mac means that you can work faster, not harder - be more productive!*

Let the original Mac Performance Shop help. Daystar has been creating Mac speed for over 16 years. Whether your bottleneck is storage, connectivity or just raw CPU speed, we deliver the performance you need, where you need it.

CPU Upgrades for Raw Speed. We upgrade any Power Macintosh, any iMac Flat Panel, any PowerBook G3 and some PowerBook G4s.

Fast and Large Storage for Real-Time Video. Our **TURBO***SATA* solutions can make your drives perform like RAM. Projects open in a flash and edit in real-time.

Extreme Wireless. Wireless is great, unless you're getting slow transfers. Even Airport Extreme's are slow when the signal is weak. Daystar can boost your signals and energize your wireless network.

But, if You Really need a G5? Daystar is the only Mac Performance Manufacturer that is also an Apple Authorized Reseller. Not only can you trade-in your system for the latest and greatest... but the Daystar Pro's can upgrade it for maximum performance!

Call 877-439-8646 and beat the clock.

Authorized Reseller

Daystar TECHNOLOGY

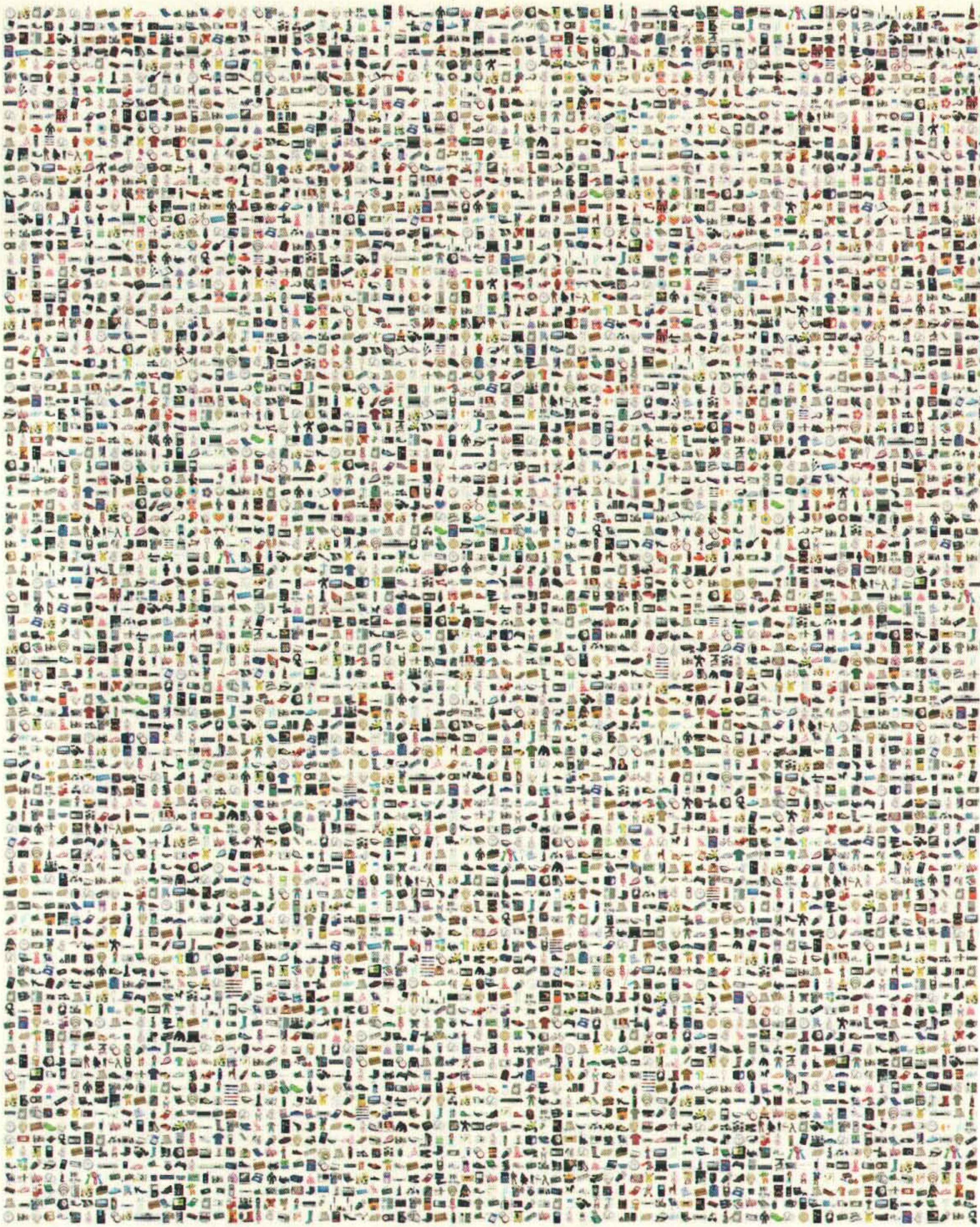
Daystar Technology - Your Macintosh Performance Shop
5018 Bristol Industrial Way, #202, Buford, GA 30518 USA
Toll Free: 877-439-8646 or 770-614-5400

Daystar-Tech.com **Daystar-Forum.com** **Daystar-Store.com**

THINGS MADE IN CHINA

COMPRESSED BY

STUFFIT DELUXE MAC/PC



DOWNLOAD AT WWW.STUFFIT.COM / COPYRIGHT © 2006 ALLUME SYSTEMS, I